GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Feature Selection and Hyper-parameter Tuning Techniques using Neural Network for stock market prediction

A Project Report of Capstone Project - 2

Submitted by

Karanveer Singh

(1613101324 / 16SCSE101850)

in partial fulfilment for the award of the degree

of

Bachelor of Technology

in

Computer Science and Engineering

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under the supervision of

Mr. Arjun K P

Assistant Professor

APRIL / MAY -2020

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"FEATURE SELECTION AND HYPER-PARAMETER TUNING TECHNIQUES USING NEURAL NETWORK FOR STOCK MARKET PREDICTION"** is the bonafide work of **"KARANVEER SINGH (1613101324)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD	SIGNATURE OF SUPERVISOR
Dr. MUNISH SHABARWAL,	**Mr. ARJUN K P,**
PhD (Management), PhD (CS)	**M.Tech (CS)**
Professor & Dean	**Assistant Professor**
School of Computer Science & Engineering	**School of Computer Science & Engineering**

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	6
1.	INTRODUCTION	7
2.	LITERATURE REVIEW	9
3.	PROPOSED SYSTEM	12
3.1	Dataset Description	12
3.2	Feature Extracting	14
3.3	Methodology	15
3.4	Performance Indicators	17
4.	IMPLEMENTATION	19
5.	RESULT	30
5.1	Hyperparameter Analysis	30
5.2	Comparative Analysis	31
5.3	Prediction csv	32
6.	CONCLUSION / FUTURE ENHANCEMENT	33
7.	REFERENCES	34

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.	Correlation between features heatmap representation	13
2.	Heat map for correlated values less than 0.9	14
3.	Visualization of selected 167 features data entries	15
4.	Plot of Neural Network Model Graph	16
5.	Hyperparameter Analysis of Proposed Method	30
6.	Comparison of actual values and predicted values at different count of ids	31
7.	Accuracy performance comparison between other models and proposed model	32
8.	Predicted probability	32

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
1.	Details of each feature in the Numerai dataset	12
2.	Performance Indicators of Regression	17

ABSTRACT

Conjecture of stock Exchange is the demonstration of attempting to decide the forecast estimation of a particular sector or the market, or the market as a whole. Every stock every investor needs to foresee the future estimation of stocks so predicted forecast of a stock's future cost could return huge benefit. To increase the accuracy of the Conjecture of stock Exchange with daily changes in the value of the market is a bottleneck task. The existing stock market prediction focused on forecasting the daily stock market by using various machine learning algorithms and deep methodologies. The proposed work we have implemented describes the new neural network model with the help of various learning techniques. The prediction of the Stock exchange is an active area for research and completion in numerai. The numerai is the toughest data science competition for stock market prediction. Numerai provides weekly new datasets to molding the finest prediction model. The dataset has 310 features, and the entries are more than 100000 per week. Our proposed new neural network model gives accuracy is closely 86%. The important point, it’s very difficult our proposed model with existing models because we are training and testing the proposed model with a new unlabelled dataset in every week. Our ultimate aim for participating in numerai competition is to suggest a neural network methodology to forecast the stock exchange independent of datasets with good accuracy.

Keyword: Deep Learning, Neural Network, Stock Market Prediction, Numerai, NMR.

1. INTRODUCTION

Stock market prediction [1] is an active area for the researcher to help in stock market prediction, we can easily determine the future values of a company stock or a financial data. So, why is stock market prediction being important, the reason is that the investors believe that the only time to invest in the market is when it is going up. When the market falls, such investors would like to stay away and return only when they are confident that the market will rise again and the better result will be available for them. Predicting the market means predicting how the stock market index moves. We can easily predict the stock market with the help of the machine learning models. In this approach we can use different model to get the better accuracy and result for the prediction. For this prediction we have used the neural network approach for the stock market prediction. We are using the real time numerai stock market dataset which is unlabelled dataset.

Numerai is basically a data science competition powers the hedge fund [2]. In this competition number of data scientist take part in the competition. The numerai competition has the similarities like we are predicting the stock a company which is not labelled. The quality of the dataset which we get form the numerai is very good and the dataset which we use it changes every week. So, we can predict good amount of data and then we can upload the prediction for checking the prediction on their model.

Neural networks are the interesting models [3] over the past years, and these models are successfully applied across an exceptional range of problem domains, the neural networks can be used in the finance, medicine, engineering, image recognition and etc. A three-layer neural network is used in the universe. There are input layer, hidden layer and output layer. These layers help in finding the result for the problem. The neural network approach takes the set of inputs (features) and computes as output as a prediction. There are other models used for prediction analysis like random forest, decision tree, support vector machine and soon. But in this paper, we have used the neural network approach because it works well for a variety of prediction problems and can easily compare the dataset. Predicting stock costs is a very important objective within the economic world [4-6], since a fairly correct prediction has the possibly to return high money edges and hedge fund risk. With the rapid increase of net and computational technology, the frequencies for acting operations on the stock prediction had increases to fractions of seconds [7-8].

The neural networks model is used to predict the stock market prediction because they are able to learn nonlinear mapping between inputs and outputs. The neural network is trained to perform a variety of financial related tasks. In numerai competition the dataset which we use for the prediction is unlabelled. With the help of neural network, it is easy to map the features which are given in the dataset. The neural networks have the ability for nonlinear function approximation and information processing which other models do not have. Neural network is well applied to the problems in which the relationships among the data are genuinely difficult and the training data sets are large enough.

2. LITERATURE REVIEW

Related works in this stock market sector, we classify the techniques which help to compute the stock market prediction issues. The first class of connected work is economic science models, which has classical economic science models for prediction. Common ways are the auto-regressive technique (AR), the moving-average-model (MA), the auto-regressive moving-average-model (ARMA), and therefore the auto-regressive-integrated-moving average (ARIMA) [9-10]. In other words, models take every new signal as a noisy linear combination from the previous signals and freelance noisy terms. However, most of them think about some sturdy assumptions with reference to the noisy terms and loss functions. The second class, we have soft computational based models. Soft computational may be a term that covers computing that mimic biological processes. These techniques embrace Neural Networks, Fuzzy Logic (FL), Support Vector machines (SVM), particle Swarm Optimization (PSO), and some other. Several researchers have tried to agitate opacity beside randomness in possibility pricing models.

Dang Lien Minh [11] has proposed the model Two-stream Gated Recurrent Unit (TGRU) for the stock market prediction in which they have used the dataset of the Reuters and Bloomberg from between October 2006 to November 2013 which they get from the yahoo finance. They conducted implementation on the NVIDIA digits toolbox with Keras API version 1.2.2 using Python version 2.7.3. The hyperparameter they used in the model was 30 epochs for training, batch size as 64, learning rate as 0.001 and learning rate decay as 0.0001. They got the overall accuracy about 66.32% on the proposed model. Yujie Wang [12] has proposed the Hybrid Time-Series Neural Network (HTPNN) model for the stock market prediction. They have used the yahoo finance dataset. In the HTPNN model they have used the 2-convolution layer, 2-LSTM layer, learning rate 0.005 and they have used 1000 iteration. Experimental result showed the model got 69.51 Accuracy.

Sheik Mohammad Idress [13] has proposed the ARIMA model for the Indian Stock market, they have used the Indian stock market data from Jan 2012 to Dec 2016 which is related to the Sensex and Nifty. There p-value for the Nifty and Sensex has 0.9099 and 0.8682. Hiransha M [14] has proposed the 4 different model for the stock prediction are Multi Layer Perception (MLP), Recurrent Neural Network (RNN), Convolution Neural network (CNN) and LSTM (Long Short-Term Memory) for the dataset from the National Stock Exchange (NSE) India from that they have used the data of the Tata motors and New York Stock Exchange (NYSE) they have used the data of Bank of America (BSC) and Chesapeak energy (CHK). There CNN model performs wells against the other 3 model. Guang Liu [15] proposed a model Numerical Based Attention (NBA) in which they have used the LSTM as a hidden layer. The LSTM encoder and decoder are set to 64 for the better result. The dataset used from the China Securities Index 300 (CSI 300) and Standard and Poor 500 (S&P 500).

Rui Ren [16] used the two approach as sentiment analysis and support vector machine for stock market. They have used the dataset from the China SSE 50 index for stock market and also for the news documents. They have used the k-fold crossvalidation and a realistic rolling window approach. The model used for the stock market is SVM model. Dharmaraja Selvamuthu [17] proposed the ANN model for the Indian Stock Market. They had used the two different dataset names as tick by tick dataset and 15-min dataset for the stock market. In which they have used the three algorithms, i.e., Bayesian Regularization, Scaled Conjugate Gradient and Levenberg-Marquardt by predicting over the data for the stock market.

Gunduz [18] used the two different models as LSTM and Regression model based on Machine Learning to predict the stock values. The dataset obtained from the yahoo finance. In Regression based model they have set the batch size to 512 and epochs to 90. For LSTM based model they dropout 0.3 and used the RMSE. The confidence score of 0.86625 for the regression-based model. Wasiat Khan [19] used the machine learning algorithm for predicting stock market via public sentiment and political situation analysis. They have used the stock market historical data from Yahoo finance and public sentiment data they have used from Twitter. They have used many algorithms like DT, SVM, RF, MLP and etc., but the two algorithm gives the better result than other algorithms. The MLP and DT gives the better result for the stock market. They achieved the accuracy up to 68%.

Catalin Stoean [20] proposed the LSTM and CNN model for the stock market. They have used the dataset of Bucharest Stock Exchange which has the data more than 20 companies for the stock market. The LSTM has the higher gain term of the CNN in the stock market. Thi-Thu Nguyen and Seokhoon Yoon [21] has proposed the DTRSI model. The DTRSI stands for Deep Transfer with Related Stock Information framework which performs well then the SVM, RF and KNN model for predicting the stock market. In this the LSTM model is used with input layer has equal the number of features and 20 time steps, two LSTM layer with 16 units and dropout to 0.5 and in the output layer it uses the one sigmoid activation unit. The dataset used is from the stock market indices, i.e., the KOSPI 200 and the S&P 500 from 31 July 2012 to 31 July2018.

The rest of the report is organized as follows. In next sections, we detail discussed about proposed method and experimental result analysis.

3. PROPOSED SYSTEM

We have performed on the proposed work in windows 10 with Intel Pentium configuration. The project was done with the help of the Google Colab server for better computational power. In the Google Colab server, we have used the GPU as the runtime for running the code efficiently. We have used some library as TensorFlow version 1.x, pandas, numpy, seaborn, matplot library, sklearn library, and Keras.

3.1 Dataset Description

The dataset which we have used in this study is from numerai, which we get is in the form of an unlabelled dataset. Numerai provided two datasets, and one is a training dataset used for training to our proposed model, and next is the testing dataset used for the testing model. The dataset which we use is changing every week. We have to upload our prediction on the numerai tournament to check the prediction is working well on their model also.

Table 1 Details of each feature in the Numerai dataset.

Variables		Class	Scale
id	Key of prediction	Categorical	Random values
era	Period of time.	Categorical	(era1, era2, era3,…,era120)
data_type	Type in the datasets	Categorical	(Train, Test, Live, Validation)
feature_intelligence1 - feature_intelligence12	Feature set 1	Numerical	(0, 0.25, 0.50, 0.75, 1)
feature_charisma1 - feature_charisma86	Feature set 2	Numerical	(0, 0.25, 0.50, 0.75, 1)
feature_strength1 - feature_strength38	Feature set 3	Numerical	(0, 0.25, 0.50, 0.75, 1)
feature_dexterity1- feature_dexterity14	Feature set 4	Numerical	(0, 0.25, 0.50, 0.75, 1)
feature_constitution1- feature_constitution114	Feature set 5	Numerical	(0, 0.25, 0.50, 0.75, 1)
feature_wisdom1- feature_wisdom46	Feature set 6	Numerical	(0, 0.25, 0.50, 0.75, 1)

target_kazutsugi	Target	Numerical	(0, 0.25, 0.50, 0.75, 1)

Table 1 shows a detailed explanation of each feature in our dataset. The training dataset has 314 columns consists of ids, eras, datatype which is train and 310 features which are subdivided into different groups like intelligence1 to intelligence12, wisdom1 to wisdom46, charisma1 to charisma86, feature dexterity1 to feature dexterity38, feature constitution1 to feature constitution114 and one target name as target_kazutsugi the values are like 0, 0.25, 0.50, 0.75 and 1 which are used to train the model. The datasets about 558069 rows with different ids and eras.

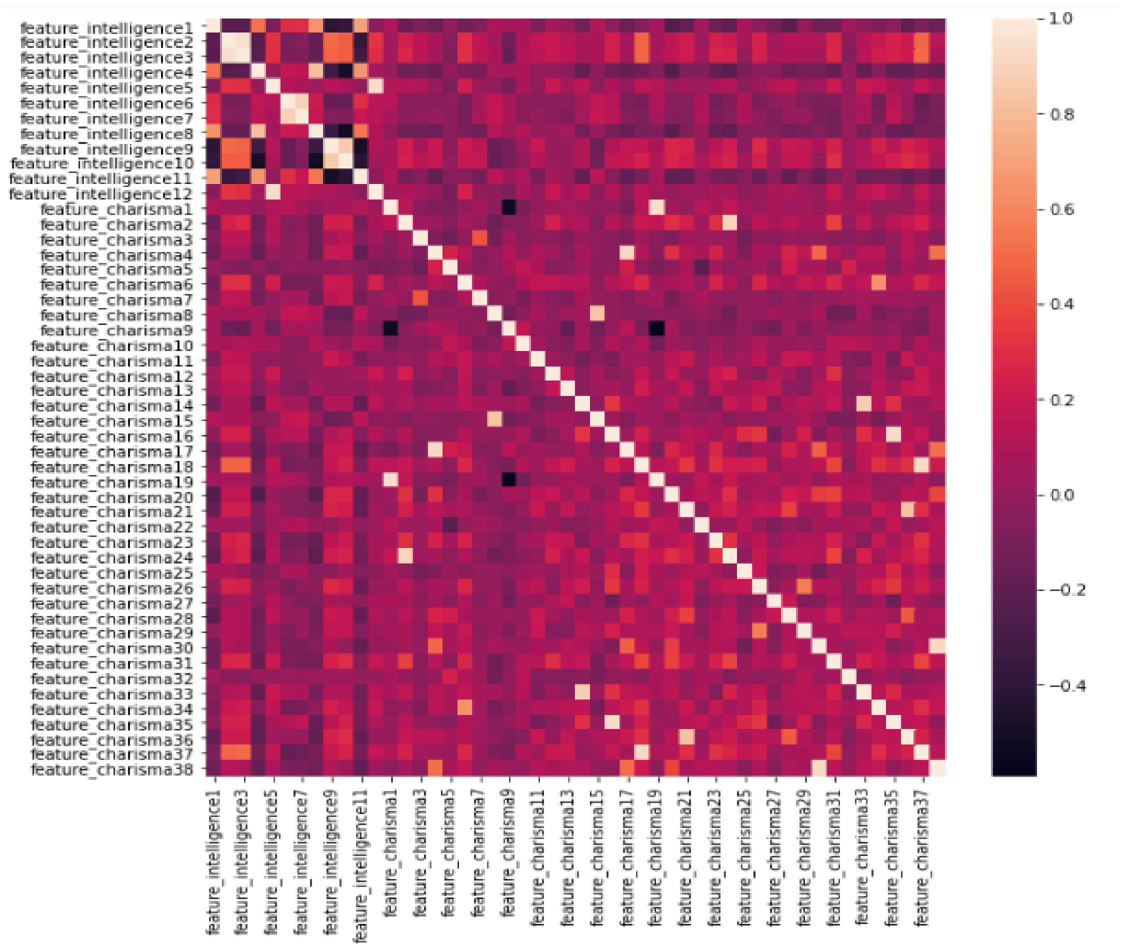


Figure 1 Correlation between features heatmap representation

Heatmap represents one variable that could be gently connected with another variable. It will be giving more effective outputs for investigations and displays more readily between factors.

Figure 1 represents the correlation between 50 features in the numerai dataset. Actually, the dataset contains 314 attributes, to represent 314 columns in the heatmap its looks messy.

The tournament dataset is our test dataset, which also has the 314 columns with the same attributes. But in the datatype, it has 3 different types as validation, test, and live. There is no need for pre-processing in this dataset because there no missing values and outliers are present.

3.2 Feature Extracting

Here the comparison of relation between features is done and remove, which has a value higher than 0.9. So, after using this, we will get the dataset, which has only those columns with a correlation of less than 0.9, as shown in figure 2.

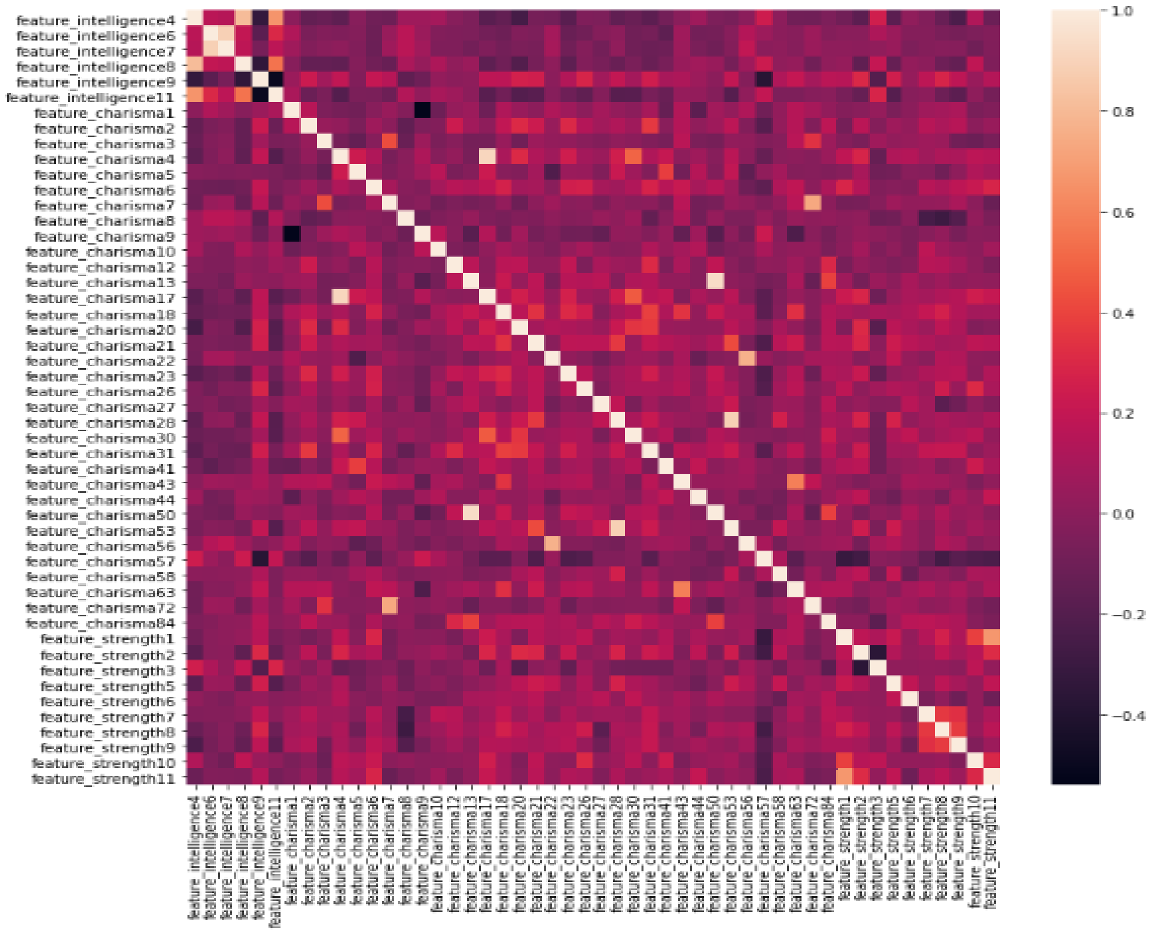


Figure 2 Heat map for correlated values less than 0.9

Now we will select the columns based on the p-value. We have built the model the makes regression and p values are calculated. If the value is higher than the fixed value, we can reject the sequence of features. The p-value which we used is 0.05. After removing the values, we can visualize the values with the help of the violin plot, as shown in figure 3.

$$P - Value < 0.05 \rightarrow Moderate\ certainty\ in\ the\ result.$$

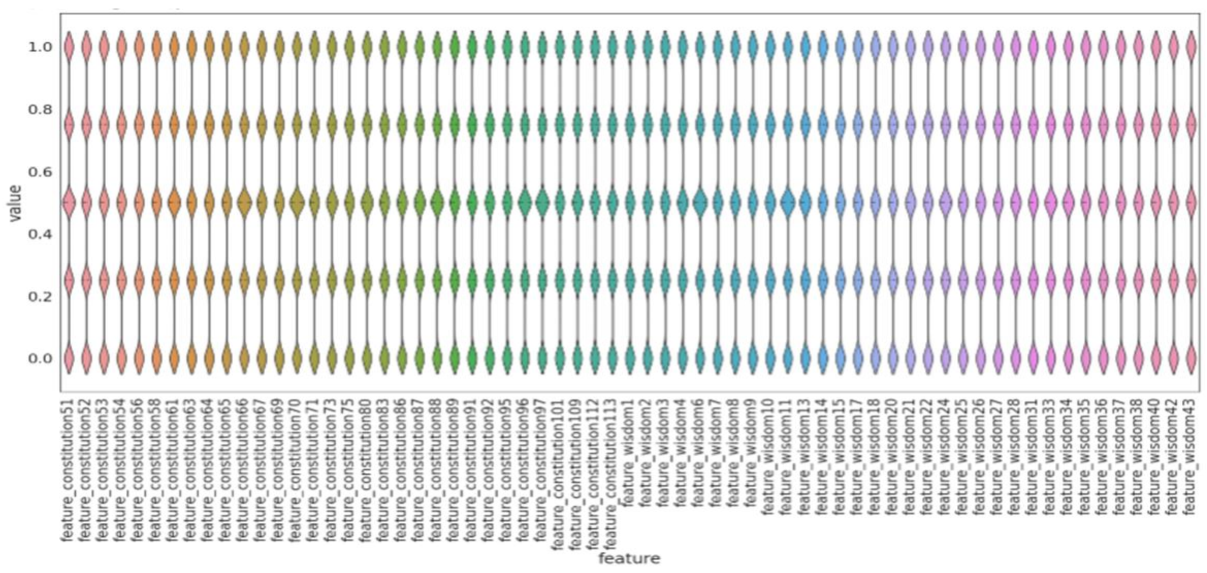


Figure 3 Visualization of selected 167 features data entries

3.3 Methodology

The neural networks have the ability to discover nonlinear relationships in input data which makes them ideal for modelling nonlinear dynamic systems such as the stock predictions. In the neural network we have used the activation function. We can use any activation function in the neural network like sigmoid, tan hyperbolic, relu, linear and radial basis function. Hidden layers and nodes in the neural network we train for the prediction of stock data has a number of hidden layers, and number of hidden nodes in each layer as shown in the figure 1.

Using Keras neural network library for python we can define neural network model which we can train on our training dataset. In neural network model we have used 400 neurons and dropout is 0.4. We have used 1 hidden layer with Relu activation function and 1 output layer with linear activation function. The linear activation function is used in the output layer because it is used for the regression problem. Then we create a wrapper for the neural network which helps to create a bridge between keras and scikit-learn. By using the keras regressor for the regression problem with the epochs 30, batch size 250 and we set the verbosity to 0 because we don't need to see how far the network has been trained. Using the RandomizedSearchCV from scikit-learn we can get good hyperparameters for the neural network model. We have tried by putting different hyperparameter which will work best. So, we have used 80 neurons or 90 neurons and dropout probability of 0.1 or 0.3. This gives a parameter with a total of 4 combinations. Then we will create the instance of RandomizedSearchCV with our model, parameter with the 4 combinations, a scoring function we have used is MSE (Mean Squared Error), one thread and a verbose level of 3. In figure 4 we can see the model which we have used the stock market prediction.

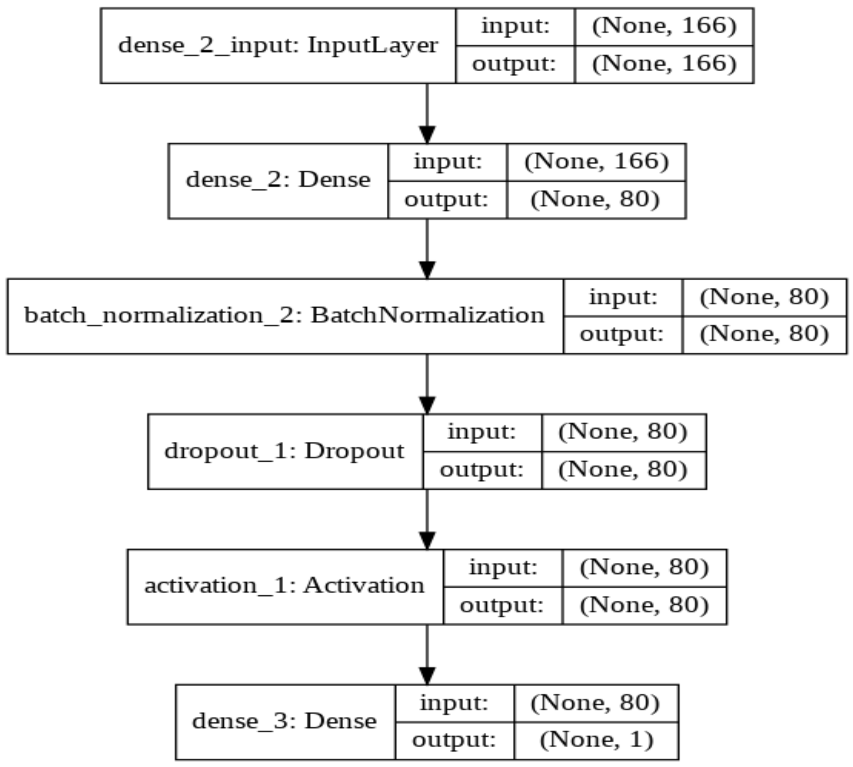


Figure 4 Plot of Neural Network Model Graph

3.4 Performance Indicators

We start by some underlying arrangement of the model and predict the output dependent on some info. The predicted value is then contrasted, and the target label and the proportion of our model execution are taken. At that point, the different parameters of the model are balanced iteratively so as to arrive at the optimal estimation of the performance metric. In most of the things, the outputs are assessed from two types: the primary is RMSE or RMSRE between real value and predicted value, the next is Mean Directional Accuracy, which suggests the proportion of correct analysis of price flow direction, as up and down movements that can extremely matter for taking any decisions. The little enhancements in prediction performance are often beneficial.

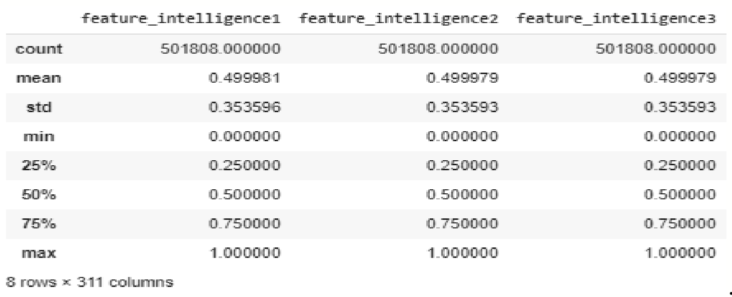
Table 2 Performance Indicators of Regression

Hyperparameter	**Explanation**
R2 score	It calculates the determination and it is for regression score function. $$\boldsymbol{R^2(y, y') = 1 - \frac{\sum_{i=n}^{n}(y_i - y'_i)^2}{\sum_{i=n}^{n}(y_i - y')^2}}$$
MAE	It finds out MAE, a parameter corresponding to the absolute error loss or l1-norm loss. $$\boldsymbol{MAE(y, y') = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples-1}} \lvert y_i - y'_i \rvert}$$
MSLE	The parameter is identified for the squared logarithmic error or loss function. $$\boldsymbol{MSLE(y, y') = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples-1}} (log_e(1 + y_i) - log_e(1 + y'_i))^2}$$
MSE	It computes parameter corresponding to the value of the squared error or loss.

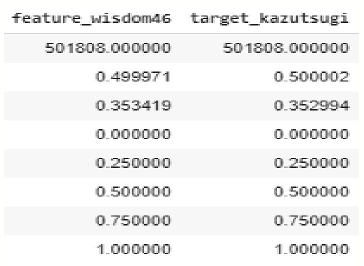
	$$\boldsymbol{MSE(y,y') = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples-1}} (y_i - y'_i)^2}$$
RMSE	It measures the standard deviation of the mistakes which occurs when a prediction is made on a dataset. $$\boldsymbol{RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}}$$
Max Error	It identifies the between forecasted and original value error. $$\boldsymbol{Max\ Error(y,y') = max(\lvert y_i - y'_i \rvert)}$$

4. IMPLEMENTATION

```
#In this step we are mounting the google drive. The dataset which is
uploaded on the drive will be easy to access.

from google.colab import drive

drive.mount('/content/drive')

#Importing the tensorflow version 1

%tensorflow_version 1.x
import tensorflow


import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline


# reading csv files
training_data = pd.read_csv('/content/drive/My Drive/dataset/numerai_tr
aining_data.csv',header=0)
tournament_data =pd.read_csv('/content/drive/My Drive/dataset/numerai_t
ournament_data.csv',header=0)


training_data.describe()
```



```
       feature_intelligence1  feature_intelligence2  feature_intelligence3
count          501808.000000          501808.000000          501808.000000
mean                0.499981               0.499979               0.499979
std                 0.353596               0.353593               0.353593
min                 0.000000               0.000000               0.000000
25%                 0.250000               0.250000               0.250000
50%                 0.500000               0.500000               0.500000
75%                 0.750000               0.750000               0.750000
max                 1.000000               1.000000               1.000000
8 rows × 311 columns
```



```
feature_wisdom46  target_kazutsugi
501808.000000     501808.000000
0.499971          0.500002
0.353419          0.352994
0.000000          0.000000
0.250000          0.250000
0.500000          0.500000
0.750000          0.750000
1.000000          1.000000
```

….

```
tournament_data.describe()
```

	feature_intelligence1	feature_intelligence2	feature_intelligence3	...	feature_wisdom46	target_kazutsugi
count	1.560303e+06	1.560303e+06	1.560303e+06	...	1.560303e+06	106895.000000
mean	4.999795e-01	4.999795e-01	4.999795e-01	...	4.999849e-01	0.499960
std	3.535881e-01	3.535881e-01	3.535881e-01	...	3.535039e-01	0.353323
min	0.000000e+00	0.000000e+00	0.000000e+00	...	0.000000e+00	0.000000
25%	2.500000e-01	2.500000e-01	2.500000e-01	...	2.500000e-01	0.250000
50%	5.000000e-01	5.000000e-01	5.000000e-01	...	5.000000e-01	0.500000
75%	7.500000e-01	7.500000e-01	7.500000e-01	...	7.500000e-01	0.750000
max	1.000000e+00	1.000000e+00	1.000000e+00	...	1.000000e+00	1.000000

8 rows × 311 columns

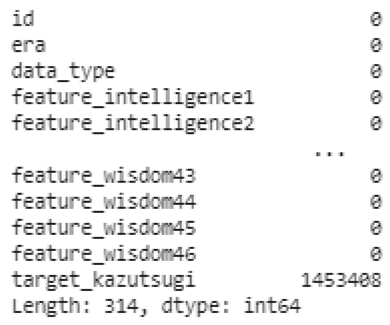
```
print(training_data.dtypes)

print(tournament_data.dtypes)
```

```
id                          object
era                         object
data_type                   object
feature_intelligence1      float64
feature_intelligence2      float64
                             ...
feature_wisdom43           float64
feature_wisdom44           float64
feature_wisdom45           float64
feature_wisdom46           float64
target_kazutsugi           float64
Length: 314, dtype: object
id                          object
era                         object
data_type                   object
feature_intelligence1      float64
feature_intelligence2      float64
                             ...
feature_wisdom43           float64
feature_wisdom44           float64
feature_wisdom45           float64
feature_wisdom46           float64
target_kazutsugi           float64
Length: 314, dtype: object
```

```
#checking that there no null value present

training_data.isnull().sum()
```

```
id                         0
era                        0
data_type                  0
feature_intelligence1      0
feature_intelligence2      0
                          ..
feature_wisdom43           0
feature_wisdom44           0
feature_wisdom45           0
feature_wisdom46           0
target_kazutsugi           0
Length: 314, dtype: int64
```

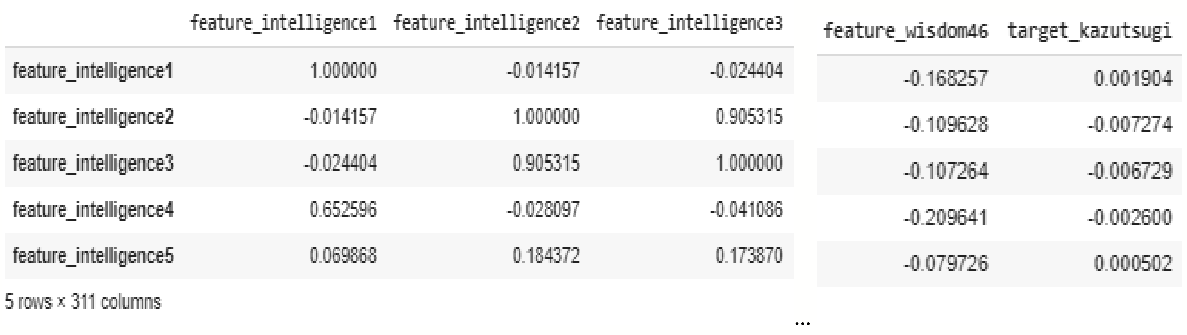
```
#As we can check that in the target value there are some null values.

For those values we have to predict.

tournament_data.isnull().sum()
```



```
id                            0
era                           0
data_type                     0
feature_intelligence1         0
feature_intelligence2         0
                         ...
feature_wisdom43              0
feature_wisdom44              0
feature_wisdom45              0
feature_wisdom46              0
target_kazutsugi        1453408
Length: 314, dtype: int64
```

```
#removing the columns which are not required for the feature scaling

data = training_data.drop(labels=['id','era','data_type'],axis=1)

data.shape

>>>(501808, 311)
```

```
#finding the correlation between the feature using correlation matrix

corr = data.corr()

corr.head()
```



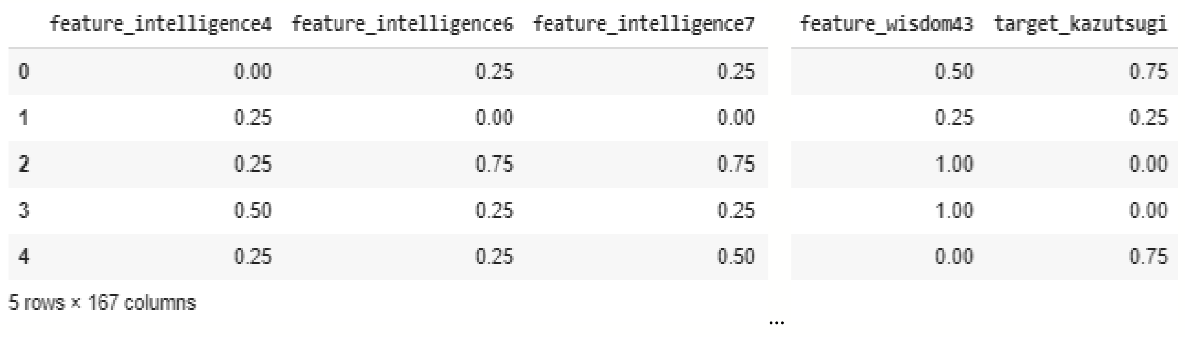
	feature_intelligence1	feature_intelligence2	feature_intelligence3	...	feature_wisdom46	target_kazutsugi
feature_intelligence1	1.000000	-0.014157	-0.024404		-0.168257	0.001904
feature_intelligence2	-0.014157	1.000000	0.905315		-0.109628	-0.007274
feature_intelligence3	-0.024404	0.905315	1.000000		-0.107264	-0.006729
feature_intelligence4	0.652596	-0.028097	-0.041086		-0.209641	-0.002600
feature_intelligence5	0.069868	0.184372	0.173870		-0.079726	0.000502

5 rows × 311 columns

```
#removing the correlated features

columns = np.full((corr.shape[0],), True, dtype=bool)

for i in range(corr.shape[0]):

    for j in range(i+1, corr.shape[0]):
        if corr.iloc[i,j] >= 0.9:
            if columns[j]:
                columns[j] = False
selected_columns = data.columns[columns]
data = data[selected_columns]

#checking the shape after removing the features
data.shape
>>> (501808, 202)

#removing more features with the help of backward elimination
selected_columns = selected_columns[1:]
import statsmodels.regression.linear_model as sm
def backwardElimination(x, Y, sl, columns):
    numVars = len(x[0])
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(Y,x).fit()
        maxVar = max(regressor_OLS.pvalues).astype(float)
        if maxVar > sl:
            for j in range(0, numVars - i):
                if (regressor_OLS.pvalues[j].astype(float) == maxVar):
                    x = np.delete(x, j, 1)
                    columns = np.delete(columns, j)

    regressor_OLS.summary()
    return x, columns
SL = 0.05
data_modeled, selected_columns = backwardElimination(data.iloc[:,1:].va
lues, data.iloc[:,0].values, SL, selected_columns)
```

```
#checking the last selected feature which we will use for training the
model
selected_columns.shape
>>>(167,)
```

```
data = pd.DataFrame(data = data_modeled, columns = selected_columns)
data.head()
```



	feature_intelligence4	feature_intelligence6	feature_intelligence7	...	feature_wisdom43	target_kazutsugi
0	0.00	0.25	0.25		0.50	0.75
1	0.25	0.00	0.00		0.25	0.25
2	0.25	0.75	0.75		1.00	0.00
3	0.50	0.25	0.25		1.00	0.00
4	0.25	0.25	0.50		0.00	0.75

5 rows × 167 columns

```
import seaborn as sn
fig, ax = plt.subplots(figsize=(20,20))
sn.heatmap(data.corr(),ax=ax)
```

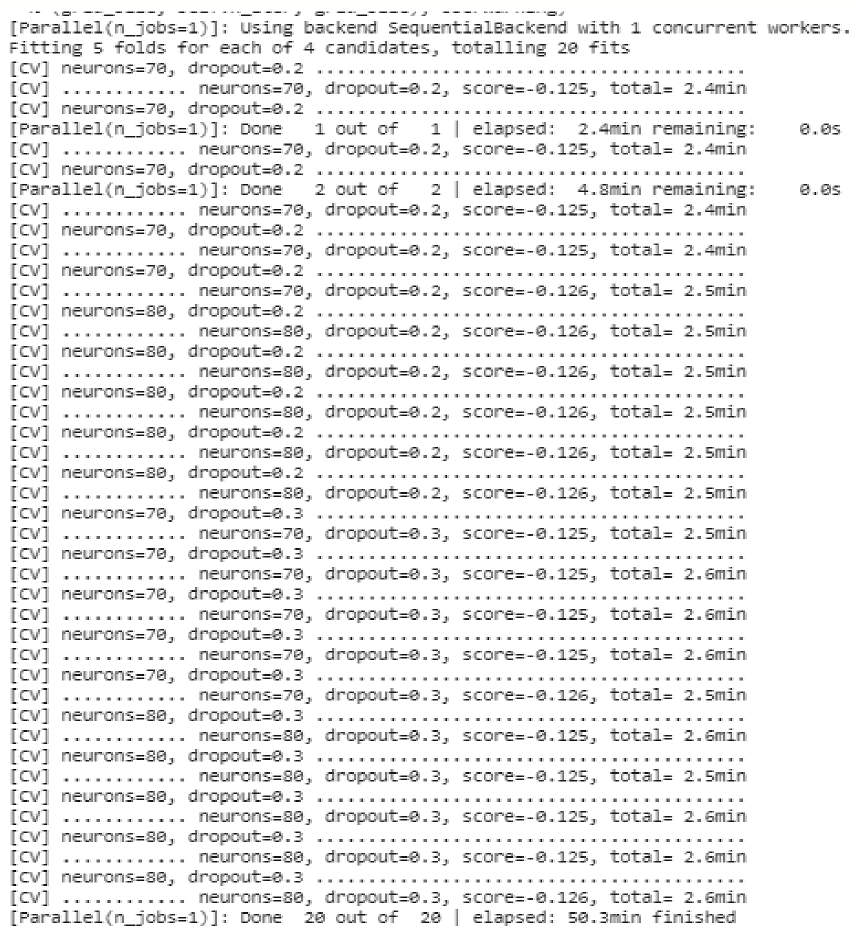
```
#Now the training datset name is data
data.shape
>>>(501808,167)
```

```
#checking the size of the testing dataset
tournament_data.shape
>>>(1560303, 314)
```

```
#making the updated dataset for the test by using the selected features
test_data = tournament_data.loc[:,selected_columns]
```

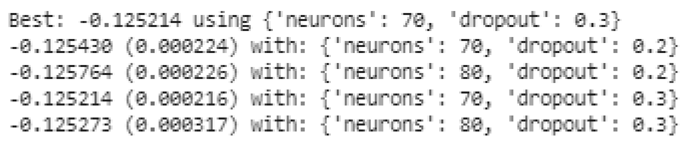
```
test_data.shape

>>>(1560303, 167)

#adding the id, era, datatype columns

n=tournament_data.loc[:,['id','era','data_type']]


data_t = pd.concat([n, test_data],axis=1)

train_data = pd.concat([training_data.iloc[:,:3], data],axis=1)


#adding validation data into the training data for training the model

validation_data = data_t[data_t.data_type=='validation']

complete_training_data = pd.concat([train_data,validation_data])


#checking the size of the validation data

validation_data.shape

>>>(106895, 170)

#training data + validation data

complete_training_data.shape

>>>(608703, 170)

features = [f for f in list(complete_training_data) if "feature" in f]

X = complete_training_data[features]

Y = complete_training_data["target_kazutsugi"]



#importing libraries for the model and the evaluation

from sklearn.metrics import log_loss

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import GroupKFold

from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, Dropout, Activation
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score as r2



#Performing Predictions with Keras and scikit-learn
def create_model(neurons=400, dropout=0.4):
    model = Sequential()
    model.add(Dense(neurons, input_shape=(310,), kernel_initializer='gl
orot_uniform', use_bias=False))
    model.add(BatchNormalization())
    model.add(Dropout(dropout))
    model.add(Activation('relu'))
    model.add(Dense(1, activation='linear', kernel_initializer='glorot_
normal'))
    model.compile(loss='mse', optimizer='adam', metrics=['mse'])
    return model
#we are using the keras regressor for building the regression model
model = KerasRegressor(build_fn=create_model, epochs=30, batch_size=400
, verbose=0)

gkf = GroupKFold(n_splits=5)
kfold_split = gkf.split(X, Y, groups=complete_training_data.era)

neurons = [70, 80]
```

```
dropout = [0.2, 0.3]

param_grid = dict(neurons=neurons, dropout=dropout)


rsearch = RandomizedSearchCV(estimator=model, param_distributions=param
_grid, n_iter=200, verbose = 3)

rsearch_result = rsearch.fit(X.values, Y.values)
```



```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Fitting 5 folds for each of 4 candidates, totalling 20 fits
[CV] neurons=70, dropout=0.2 .........................................
[CV] ............ neurons=70, dropout=0.2, score=-0.125, total= 2.4min
[CV] neurons=70, dropout=0.2 .........................................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:  2.4min remaining:    0.0s
[CV] ............ neurons=70, dropout=0.2, score=-0.125, total= 2.4min
[CV] neurons=70, dropout=0.2 .........................................
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:  4.8min remaining:    0.0s
[CV] ............ neurons=70, dropout=0.2, score=-0.125, total= 2.4min
[CV] neurons=70, dropout=0.2 .........................................
[CV] ............ neurons=70, dropout=0.2, score=-0.125, total= 2.4min
[CV] neurons=70, dropout=0.2 .........................................
[CV] ............ neurons=70, dropout=0.2, score=-0.126, total= 2.5min
[CV] neurons=80, dropout=0.2 .........................................
[CV] ............ neurons=80, dropout=0.2, score=-0.126, total= 2.5min
[CV] neurons=80, dropout=0.2 .........................................
[CV] ............ neurons=80, dropout=0.2, score=-0.126, total= 2.5min
[CV] neurons=80, dropout=0.2 .........................................
[CV] ............ neurons=80, dropout=0.2, score=-0.126, total= 2.5min
[CV] neurons=80, dropout=0.2 .........................................
[CV] ............ neurons=80, dropout=0.2, score=-0.126, total= 2.5min
[CV] neurons=80, dropout=0.2 .........................................
[CV] ............ neurons=80, dropout=0.2, score=-0.126, total= 2.5min
[CV] neurons=70, dropout=0.3 .........................................
[CV] ............ neurons=70, dropout=0.3, score=-0.125, total= 2.5min
[CV] neurons=70, dropout=0.3 .........................................
[CV] ............ neurons=70, dropout=0.3, score=-0.125, total= 2.6min
[CV] neurons=70, dropout=0.3 .........................................
[CV] ............ neurons=70, dropout=0.3, score=-0.125, total= 2.6min
[CV] neurons=70, dropout=0.3 .........................................
[CV] ............ neurons=70, dropout=0.3, score=-0.125, total= 2.6min
[CV] neurons=70, dropout=0.3 .........................................
[CV] ............ neurons=70, dropout=0.3, score=-0.126, total= 2.5min
[CV] neurons=80, dropout=0.3 .........................................
[CV] ............ neurons=80, dropout=0.3, score=-0.125, total= 2.6min
[CV] neurons=80, dropout=0.3 .........................................
[CV] ............ neurons=80, dropout=0.3, score=-0.125, total= 2.5min
[CV] neurons=80, dropout=0.3 .........................................
[CV] ............ neurons=80, dropout=0.3, score=-0.125, total= 2.6min
[CV] neurons=80, dropout=0.3 .........................................
[CV] ............ neurons=80, dropout=0.3, score=-0.125, total= 2.6min
[CV] neurons=80, dropout=0.3 .........................................
[CV] ............ neurons=80, dropout=0.3, score=-0.126, total= 2.6min
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 50.3min finished
```

```
print("Best: %f using %s" % (rsearch_result.best_score_, rsearch_result
.best_params_))

means = rsearch_result.cv_results_['mean_test_score']

stds = rsearch_result.cv_results_['std_test_score']

params = rsearch_result.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):

    print("%f (%f) with: %r" % (mean, stdev, param))
```



```
Best: -0.125214 using {'neurons': 70, 'dropout': 0.3}
-0.125430 (0.000224) with: {'neurons': 70, 'dropout': 0.2}
-0.125764 (0.000226) with: {'neurons': 80, 'dropout': 0.2}
-0.125214 (0.000216) with: {'neurons': 70, 'dropout': 0.3}
-0.125273 (0.000317) with: {'neurons': 80, 'dropout': 0.3}
```

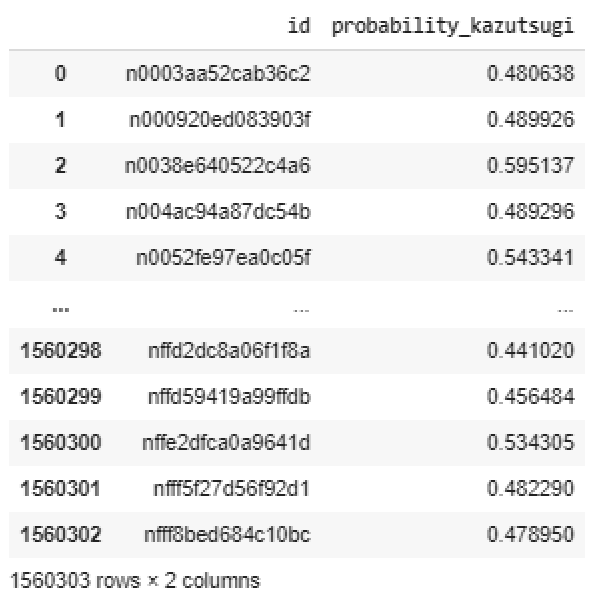
```
#Checking the Performance

def check_consistency(model, valid_data):
    eras = valid_data.era.unique()
    count = 0
    count_consistent = 0
    for era in eras:
        count += 1
        current_valid_data = valid_data[validation_data.era==era]
        features = [f for f in list(complete_training_data) if "feature
" in f]
        X_valid = current_valid_data[features]
        Y_valid = current_valid_data["target_kazutsugi"]
        loss = model.evaluate(X_valid.values, Y_valid.values, batch_siz
e=250, verbose=3)[0]
        if (loss < -np.log(.5)):
            consistent = True
            count_consistent += 1
        else:
            consistent = False
        print("{}: loss -
 {} consistent: {}".format(era, loss, consistent))
    print ("Consistency: {}".format(count_consistent/count))

check_consistency(rsearch_result.best_estimator_.model, validation_data
)
```

```
era121: loss - 0.12247831237712366 consistent: True
era122: loss - 0.12258653636076462 consistent: True
era123: loss - 0.1216976384933974 consistent: True
era124: loss - 0.1220194832713506 consistent: True
era125: loss - 0.1218868111291557 consistent: True
era126: loss - 0.12261844281264728 consistent: True
era127: loss - 0.12285408167861428 consistent: True
era128: loss - 0.1222559602495614 consistent: True
era129: loss - 0.12315131397670438 consistent: True
era130: loss - 0.12193600719996718 consistent: True
era131: loss - 0.12206526105588138 consistent: True
era132: loss - 0.12110913001118555 consistent: True
era197: loss - 0.121684776345489 consistent: True
era198: loss - 0.12261264658589316 consistent: True
era199: loss - 0.12404769004772437 consistent: True
era200: loss - 0.12294684866997649 consistent: True
era201: loss - 0.12238681528328113 consistent: True
era202: loss - 0.12233108803190745 consistent: True
era203: loss - 0.1230597155014736 consistent: True
era204: loss - 0.12218870022131698 consistent: True
era205: loss - 0.12278868713465062 consistent: True
era206: loss - 0.12241374764279264 consistent: True
Consistency: 1.0
```

```
#Submitting the Predictions

import time

x_prediction = data_t[features]

t_id = data_t["id"]

y_prediction = rsearch_result.best_estimator_.model.predict_proba(x_pre

diction.values, batch_size=400)

results = np.reshape(y_prediction,-1)

results_df = pd.DataFrame(data={'probability_kazutsugi':results})



y_prediction
```

```
array([[0.48063824],
       [0.4899264 ],
       [0.59513664],
       ...,
       [0.53430474],
       [0.48229036],
       [0.47894982]], dtype=float32)
```

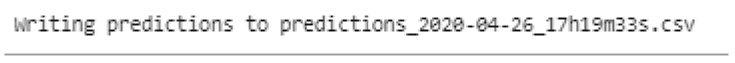
```
joined = pd.DataFrame(t_id).join(results_df)

print(joined)
```



	id	probability_kazutsugi
0	n0003aa52cab36c2	0.480638
1	n000920ed083903f	0.489926
2	n0038e640522c4a6	0.595137
3	n004ac94a87dc54b	0.489296
4	n0052fe97ea0c05f	0.543341
...	...	...
1560298	nffd2dc8a06f1f8a	0.441020
1560299	nffd59419a99ffdb	0.456484
1560300	nffe2dfca0a9641d	0.534305
1560301	nfff5f27d56f92d1	0.482290
1560302	nfff8bed684c10bc	0.478950

1560303 rows × 2 columns

```
from time import gmtime, strftime

strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

joined = pd.DataFrame(t_id).join(results_df)

# path = "predictions_w_loss_0_" + '{:4.0f}'.format(history.history['lo

ss'][-1]*10000) + ".csv"

path = 'predictions_{:}'.format(strftime("%Y-%m-

%d_%Hh%Mm%Ss", time.gmtime())) + '.csv'

print()

print("Writing predictions to " + path.strip())

# # Save the predictions out to a CSV file

joined.to_csv(path,float_format='%.15f', index=False)
```



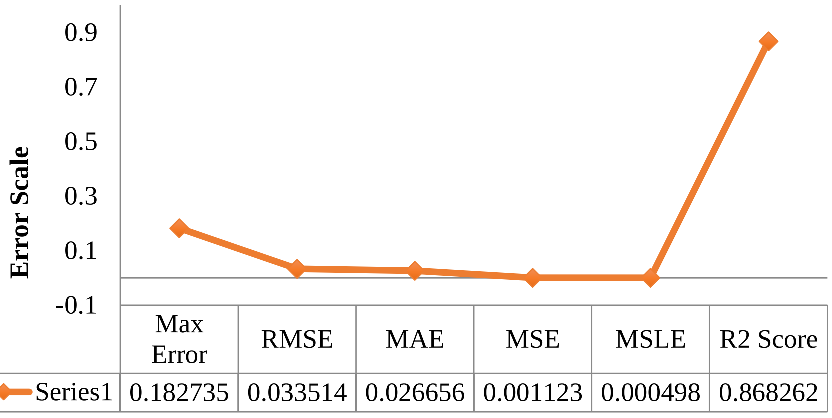
```
Writing predictions to predictions_2020-04-26_17h19m33s.csv
```

5. EXPERIMENTAL RESULTS

In numerai competition, we can get the results by uploading our prediction on the numerai tournament. Numerai measures Performance-based on the correlation of rank and predictions and actual targets. By correlation matrix, we can show the heatmap for the features, as shown in figure 1. By this, we can quickly check the data and some of the features which are related to others.

5.1 Hyperparameter Analysis

In the figure 5 graph we show the values of error and the score for the model. In the regression problem we use the MSE, MAE, Max Error, RMSE and MSLE for checking the error we get for the predictions. With the R2 score we can check the model is fit properly. If the value is less than 0.5, we can assume that the model is poorly fit. So, as the R2 score is close to 1 the model is fitted properly.



	Max Error	RMSE	MAE	MSE	MSLE	R2 Score
Series1	0.182735	0.033514	0.026656	0.001123	0.000498	0.868262

Figure 5 Hyperparameter Analysis of Proposed Method

Next, we compared the prediction with the actual prediction to check that they are near to each other. The prediction depends upon the R2 score, as the R2 score is near to 1 the values will us good. So, with the help of figure 6 graph we can easily check the values are close to each other.

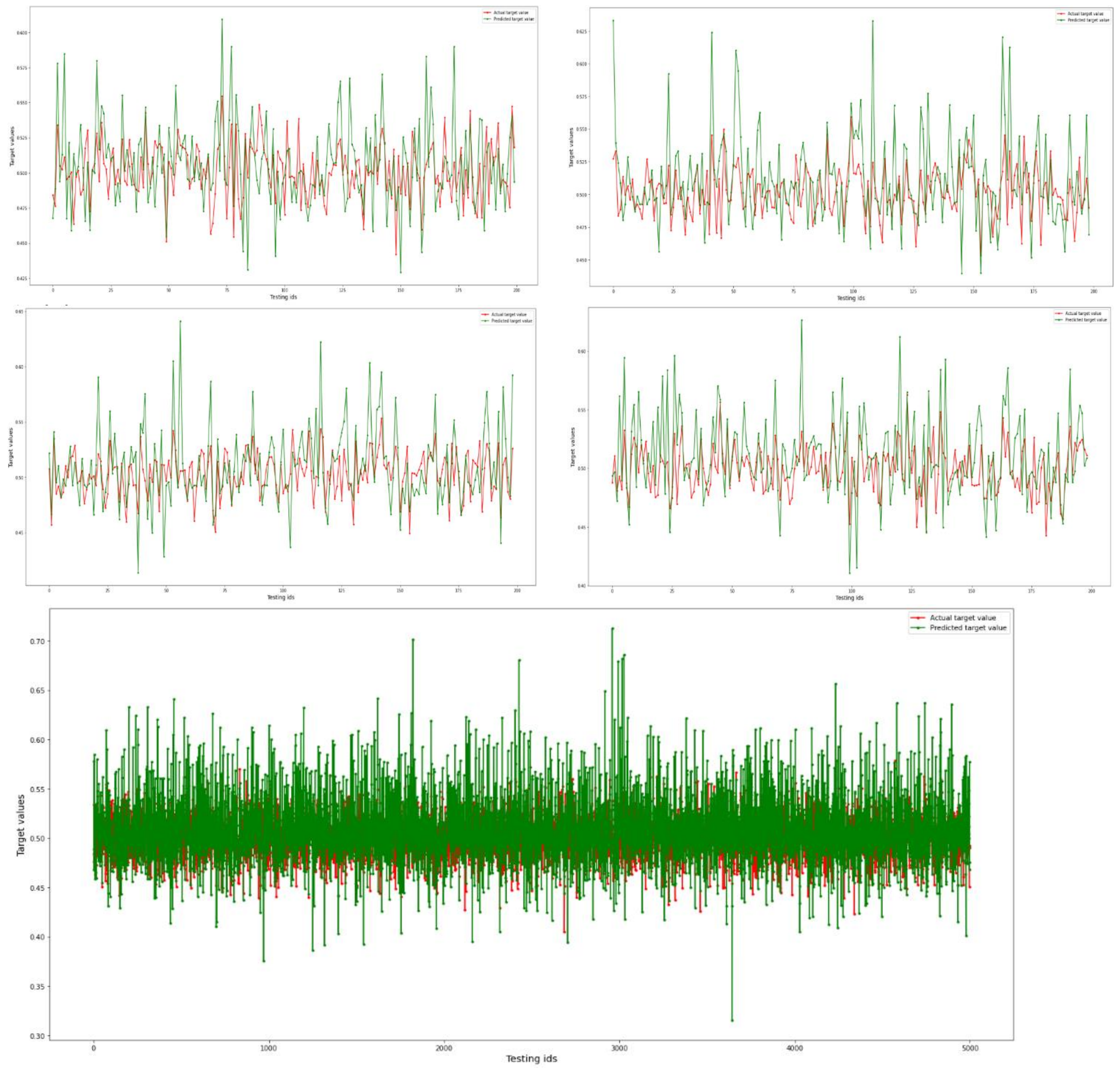


Figure 6 Comparison of actual values and predicted values at different count of ids

5.2 Comparative Analysis

In this section we compare our model with existing models and figure 7 shows accuracy performance of our proposed model. The comparison graph clearly shows our proposed method accuracy is high compared to all the previous works.

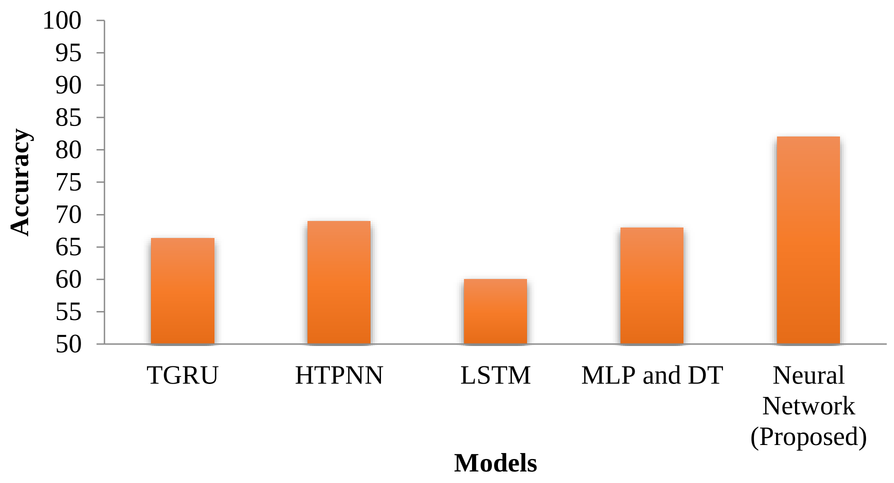
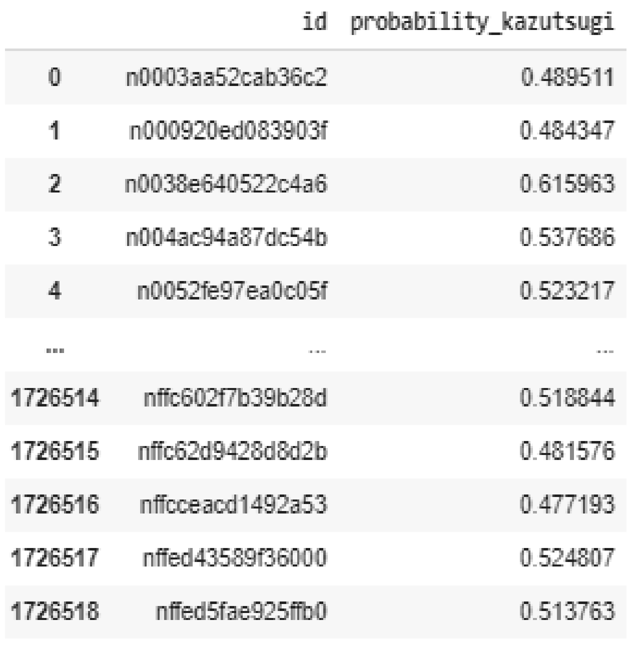


Figure 7 Accuracy performance comparison between other models and proposed model

5.3 Prediction csv

In the prediction csv file, we have two columns which are id and probability kazutsugi as shown in the figure 8. This file we will upload on the numerai tournament to check the result of the prediction.



	id	probability_kazutsugi
0	n0003aa52cab36c2	0.489511
1	n000920ed083903f	0.484347
2	n0038e640522c4a6	0.615963
3	n004ac94a87dc54b	0.537686
4	n0052fe97ea0c05f	0.523217
...	...	...
1726514	nffc602f7b39b28d	0.518844
1726515	nffc62d9428d8d2b	0.481576
1726516	nffcceacd1492a53	0.477193
1726517	nffed43589f36000	0.524807
1726518	nffed5fae925ffb0	0.513763

Figure 8 Predicted probability

6. CONCLUSION

In our proposed work we used customized Neural Network model of deep learning techniques for the prediction of the stock market value. An experimental result shows that the proposed neural network gave the good predictions for the targets and the consistency of the model is also good. Adaptive nature of Neural Network enables to make connections between input and output values in such a way that generated network becomes capable to predict the expected trends in stock market for future. Hence, we can see that Neural Networks are the efficient model for stock market prediction and can be used on real time datasets. The proposed neural network model experimental result shows the high stock market prediction accuracy that is 86% accuracy on the training set as well as the 14% loss on the testing set compared with all the existing models. Our future work is to implement an efficient neural network model to increase the performance of the stock market prediction.

7. REFERENCE

[1] Yuan, X., Yuan, J., Jiang, T., & Ain, Q. U. (2020). Integrated Long-term Stock Selection Models Based on Feature Selection and Machine Learning Algorithms for China Stock Market. IEEE Access, 1–1. doi:10.1109/access.2020.2969293

[2] Lee, J., Kim, R., Koh, Y., & Kang, J. (2019). *Global Stock Market Prediction Based on Stock Chart Images Using Deep Q-Network. IEEE Access, 1–1.* doi:10.1109/access.2019.2953542

[3] Parmar, I., Agarwal, N., Saxena, S., Arora, R., Gupta, S., Dhiman, H., & Chouhan, L. (2018). *Stock Market Prediction Using Machine Learning. 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC).* doi:10.1109/icsccc.2018.8703332

[4] R. Al-Hmouz and A. Balamash, "Description and prediction of time series: a general framework of Granular Computing," Expert Systems with Applications, vol. 42, no. 10, pp. 4830–4839, 2015.

[5] [Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. Foundations and trendsR in Machine Learning, 2(1):1–127, 2009.

[6] A. Bagheri, H. Mohammadi Peyhani, and M. Akbari, "Financial forecasting using ANFIS networks with Quantum-behaved Particle Swarm Optimization," Expert Systems with Applications, vol. 41, no. 14, pp. 6235–6250, 2014.

[7] Y. Son, D.-J. Noh, and J. Lee, "Forecasting trends of high-frequency KOSPI200 index data using learning classifiers," Expert Systems with Applications, vol. 39, no. 14, pp. 11607–11615, 2012.

[8] T. Kimoto, K. Asakawa,M. Yoda,M. Takeoka. "Stock market prediction system with modular neural network". Proceedings of the International Joint Conference on Neural Networks, 1990, pages 1–6

[9] W. P. Risk, G. S. Kino, and H. J. Shaw, "Fiber-optic frequency shifter using a surface acoustic wave incident at an oblique angle," Opt. Lett., vol. 11, no. 2, pp. 115–117, Feb. 1986.

[10] Stephane Lathuili ´ ere , et.al., "A Comprehensive Analysis of Deep Regression", IEEE, 2018.

[11] Dang, L. M., Sadeghi-Niaraki, A., Huynh, H. D., Min, K., & Moon, H. (2018). *Deep Learning Approach for Short-Term Stock Trends Prediction based on Two-stream Gated Recurrent Unit Network. IEEE Access, 1–1.* doi:10.1109/access.2018.2868970

[12] Wang, Y., Liu, H., Guo, Q., Xie, S., & Zhang, X. (2019). Stock Volatility Prediction by Hybrid Neural Network. IEEE Access, 1–1. doi:10.1109/access.2019.2949074

[13] Idrees, S. M., Alam, M. A., & Agarwal, P. (2019). A Prediction Approach for Stock Market Volatility Based on Time Series Data. IEEE Access, 1–1. doi:10.1109/access.2019.2895252

[14] M, H., E.A., G., Menon, V. K., & K.P., S. (2018). *NSE Stock Market Prediction Using Deep-Learning Models. Procedia Computer Science, 132, 1351–1362.* doi:10.1016/j.procs.2018.05.050

[15] GUANG LIU and XIAOJIE WANG, "A Numerical-based Attention Method for Stock Market Prediction with Dual Information", IEEE, 2016

[16] Ren, R., Wu, D. D., & Liu, T. (2018). *Forecasting Stock Market Movement Direction Using Sentiment Analysis and Support Vector Machine. IEEE Systems Journal, 1–11.* doi:10.1109/jsyst.2018.2794462

[17] Selvamuthu, D., Kumar, V., & Mishra, A. (2019). *Indian stock market prediction using artificial neural networks on tick data. Financial Innovation, 5(1).* doi:10.1186/s40854-019-0131-7

[18] Gunduz, H., Cataltepe, Z., & Yaslan, Y. (2017). Stock market direction prediction using deep neural networks. 2017 25th Signal Processing and Communications Applications Conference (SIU). doi:10.1109/siu.2017.7960512

[19] Khan, W., Malik, U., Ghazanfar, M. A., Azam, M. A., Alyoubi, K. H., & Alfakeeh, A. S. (2019). *Predicting stock market trends using machine learning algorithms via public sentiment and political situation analysis. Soft Computing.* doi:10.1007/s00500-019-04347-y

[20] Stoean, C., Paja, W., Stoean, R., & Sandita, A. (2019). Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations. PLOS ONE, 14(10), e0223593. doi:10.1371/journal.pone.0223593

[21] Thi-Thu Nguyen and Seokhoon Yoon,"A Novel Approach to Short-Term Stock Price Movement Prediction using Transfer Learning", Applied Sciences- MDPI, 2019.