[Stock Price Prediction]

A Project Report

In the partial fulfillment for the award of the degree of

B.Tech

under

Academy of Skill Development

Submitted by

[Jojangandha Saha]

[Heritage Institute of Technology]

STOCK PRICE PREDICTION
USING LINEAR REGRESSION

Introduction

This Stock price prediction project aims to predict future stock prices and its nature of a given stock using linear regression. Linear regression is a statistical method used to model the relationship between a dependent variable (stock prices) and one or more independent variables (features). The project utilizes historical stock price data and relevant financial indicators such as date, close price etc. as input features to build a linear regression model. The resulting model can be employed to make predictions about the stock's future prices, enabling investors and traders to make informed decisions according to the relevancy of predictions..

Objectives

The primary objectives of this project are given below::

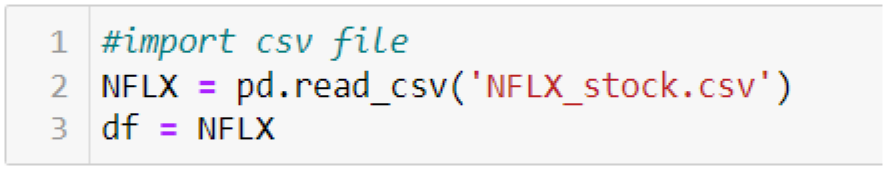
- Gather historical stock price data for the chosen stock (Kaggle is being used to collect datasets).
- Identify and select relevant financial indicators as features for prediction (Here close price is being chosen as a feature to predict upon).
- Preprocess and clean the data to handle missing values and outliers.
- Split the data into training and testing sets.
- Build a linear regression model based on the training data.
- Evaluate the model's performance using appropriate metrics.
- Cross validation is used to evaluate the best model by training and testing several other models to get the closest and minimized prediction.
- Hyperparameter tuning used to train different hyperparameter configurations for models.
- Use the trained model to predict future stock prices.

- Visualize the results for better interpretation.

Data Collection

Data collection involves obtaining historical stock price data and relevant financial indicators for the stock of interest. This stock price dataset was taken from Kaggle and the dataset was of Netflix Stock price data.

Using the code below the data has been read from dataset



```
#import csv file
NFLX = pd.read_csv('NFLX_stock.csv')
df = NFLX
```

Data Preprocessing

Data preprocessing is a crucial step to ensure the data is suitable for modeling. The following tasks will be performed during this phase:

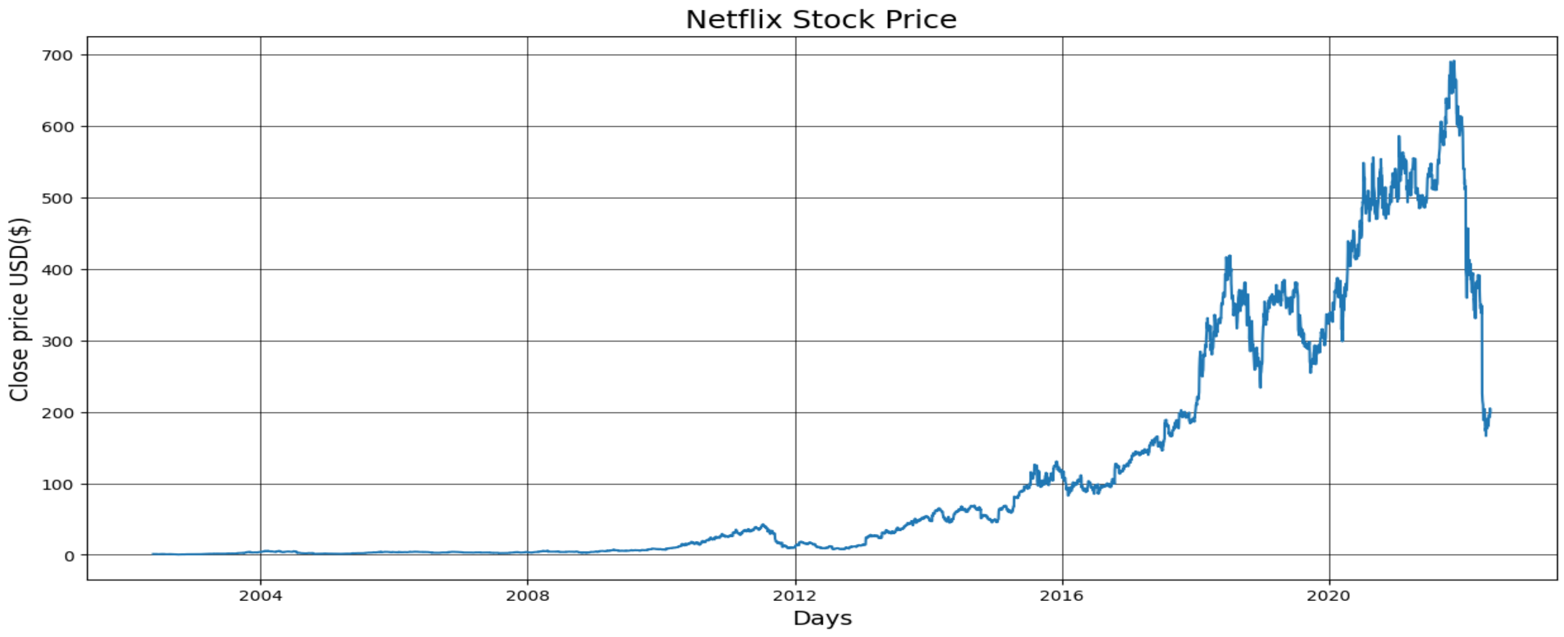
Handling missing values: Apply imputation techniques to fill missing data points or consider removing the corresponding data instances.

Outlier detection and treatment: Identify and address any outliers that may negatively impact the model's performance. Outliers may be handled by replacing them with appropriate values or removing them.

Feature engineering: Select the most relevant financial indicators and calculate any derived features that might enhance the model's predictive power. For instance, we may compute moving averages and other technical indicators to capture trends and patterns.

Scaling: Normalize or standardize the feature values to bring them to a similar scale. This step is particularly crucial when features have different units or magnitude.

Below graph represents the Close price data. Actual trend of Close Price data. We want to make sure the predicted trend will be close to similar to this graph, so that the difference between Actual and Predicted Price will be minimized and therefore can be said that the prediction was correct and close to actual price trend.



Model Building

In this project, a linear regression model will be used to predict stock prices. **Linear regression aims to find the best-fitting line** (in this case, a hyperplane in multidimensional space) to the data, which can be used for future predictions.

The model will be built using the training dataset, which consists of historical stock prices and the corresponding financial indicators. We will use the **scikit-learn library** in **Python to implement the linear regression model.**

Below code shows how the model building has been done by creating an instance of the LinearRegression() class of sklearn.linear_model

```python
# create an instance of the Lr class and send training data to it
lr = LinearRegression()
lr.fit(X_train, y_train)
```

K-Fold Cross Validation

K-Fold Cross-Validation is a technique used to evaluate the performance of a machine learning model while mitigating the risk of overfitting. It involves splitting the dataset into K subsets (or "folds") of roughly equal size. The model is trained K times, each time using K-1 of the folds as the training set and the remaining fold as the validation set. This process is repeated for each fold, and the performance metrics are averaged to obtain a more robust evaluation.

In our stock price prediction project, K-Fold Cross-Validation will be applied to assess the model's generalization capabilities on different subsets of the data. The following steps will be implemented:

Split the dataset into K subsets (usually 5 or 10) while maintaining the temporal order of the data to preserve the time-series nature of the problem.

For each iteration, train the linear regression model on K-1 subsets and evaluate it on the held-out subset.Record the evaluation metrics (e.g., MSE, R^2) for each iteration.

Below code shows, two kinds of model is being used on is DecisionTreeRegressor and RnadomForestRegressor

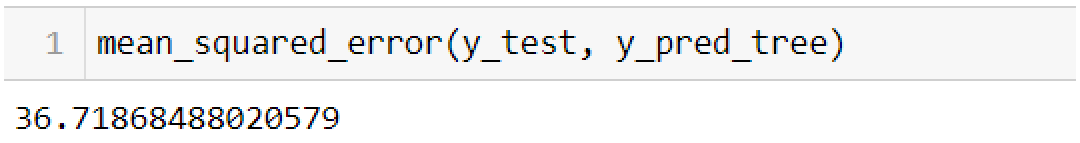
```python
#but this became overfitting model
#therefore k cross validation folds, cv = 8 means dividing into 8 folds

rmses = np.sqrt(-cross_val_score(tree_reg, X_train,y_train,cv=8,scoring="neg_mean_squared_error"))
```

```python
rmses
```

```
array([4.60403295, 8.15524873, 7.72494859, 5.27098197, 4.65782559,
       5.13572593, 7.90984736, 8.26677692])
```

```python
from sklearn.ensemble import RandomForestRegressor
rand_reg = RandomForestRegressor()
rmses = np.sqrt(-cross_val_score(rand_reg,X_train,y_train,cv=8,scoring='neg_mean_squared_error'))
```

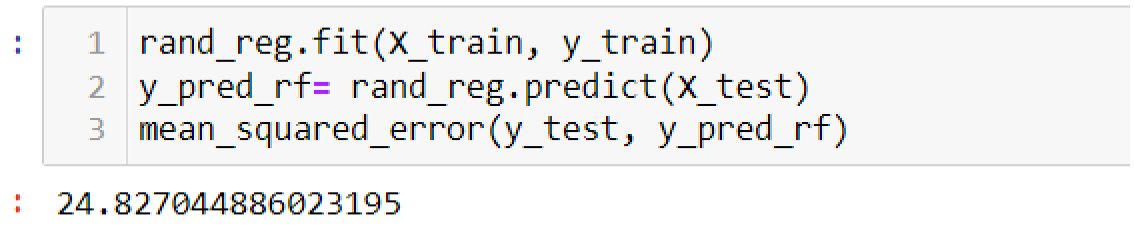
For DecisionTreeRegressor



```
mean_squared_error(y_test, y_pred_tree)
```

```
36.71868488020579
```

For RandomForestRegressor



```
rand_reg.fit(X_train, y_train)
y_pred_rf= rand_reg.predict(X_test)
mean_squared_error(y_test, y_pred_rf)
```

```
24.827044886023195
```

From the above code we can see that, minimum value for mean squared error is obtained by using a random forest regressor, now we can hyper-tune parameters of the model to obtain best results.

Hyperparameter Tuning For Model selection

Hyperparameter tuning involves finding the optimal values for hyperparameters, which are parameters set before the model training begins and cannot be learned during the training process itself. For linear regression, hyperparameters could include regularization strength, learning rate (if using gradient descent), and feature selection techniques.

In our project, we will explore hyperparameter tuning to enhance the model's performance. The following steps will be followed:

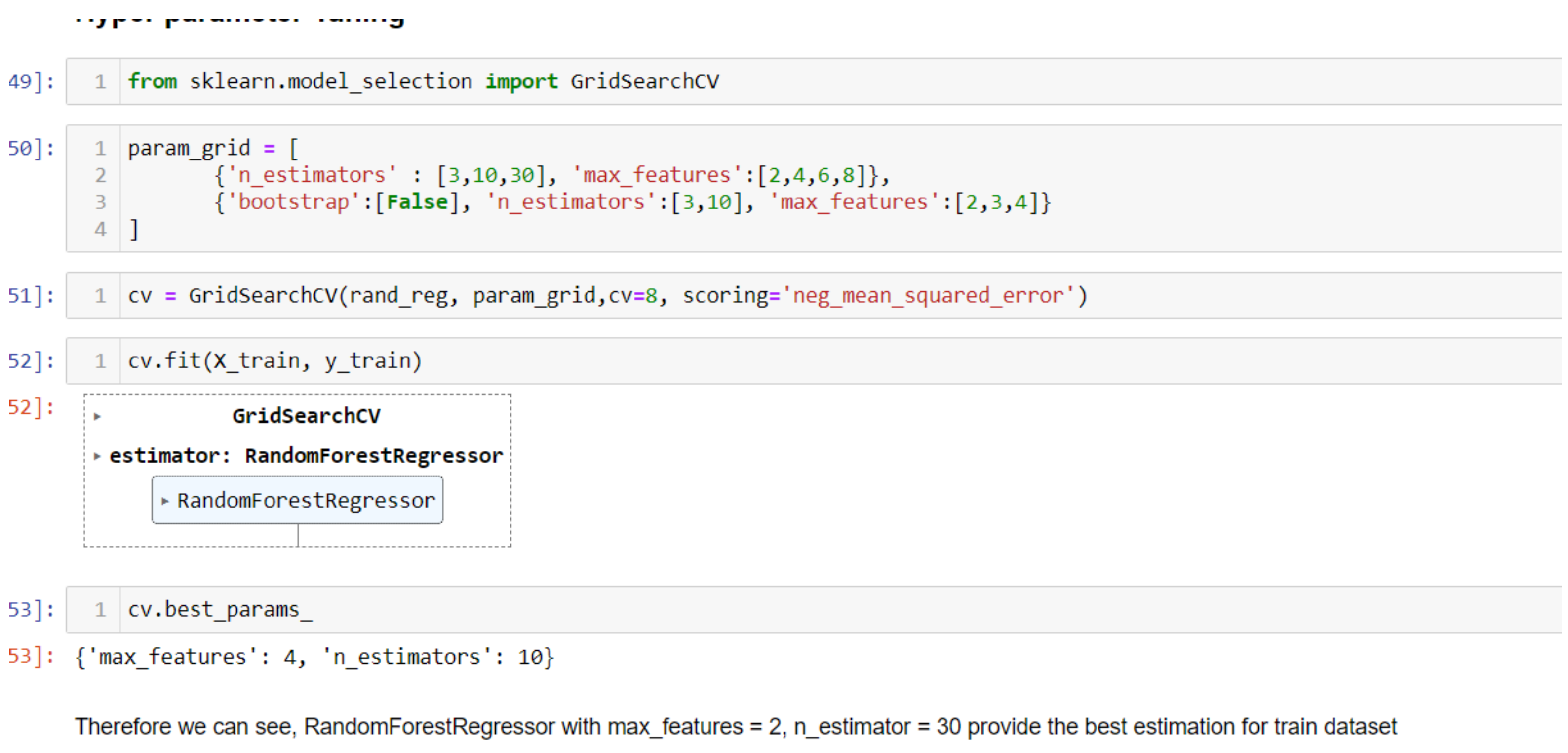
Define a range of hyperparameter values to explore for each hyperparameter of the linear regression model.

Use K-Fold Cross-Validation with different combinations of hyperparameter values to evaluate the model's performance for each setting.

Select the hyperparameter values that result in the best-performing model based on the evaluation metrics (e.g., lowest MSE or highest R^2).

To efficiently perform hyperparameter tuning, techniques such as Grid Search or Random Search can be employed. Grid Search exhaustively searches all possible combinations of hyperparameter values from predefined grids, while Random Search samples random combinations within the defined ranges.

Hyperparameter tuning will enable us to identify the best configuration for the linear regression model, leading to more accurate stock price predictions.



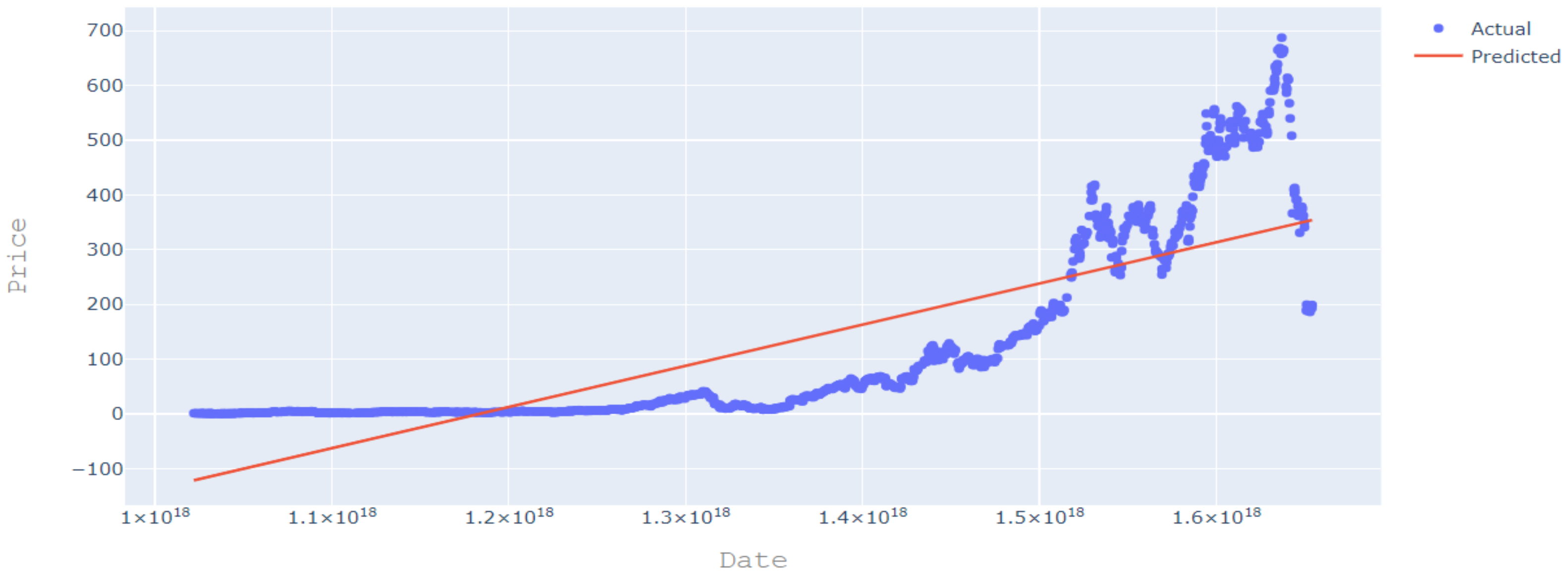
```
49]:  from sklearn.model_selection import GridSearchCV

50]:  param_grid = [
              {'n_estimators' : [3,10,30], 'max_features':[2,4,6,8]},
              {'bootstrap':[False], 'n_estimators':[3,10], 'max_features':[2,3,4]}
      ]

51]:  cv = GridSearchCV(rand_reg, param_grid,cv=8, scoring='neg_mean_squared_error')

52]:  cv.fit(X_train, y_train)

52]:  ▸ GridSearchCV
      ▸ estimator: RandomForestRegressor
          ▸ RandomForestRegressor

53]:  cv.best_params_

53]:  {'max_features': 4, 'n_estimators': 10}
```

Therefore we can see, RandomForestRegressor with max_features = 2, n_estimator = 30 provide the best estimation for train dataset

and n_estimator = 30 provides the best estimation for train datasets.

Model Prediction

After the model is trained, it will be ready for making future predictions. Historical data for the most recent period will be used as input to predict future stock prices. And the prediction will be upon the feature close price. And we want the prediction to be close to the actual values.

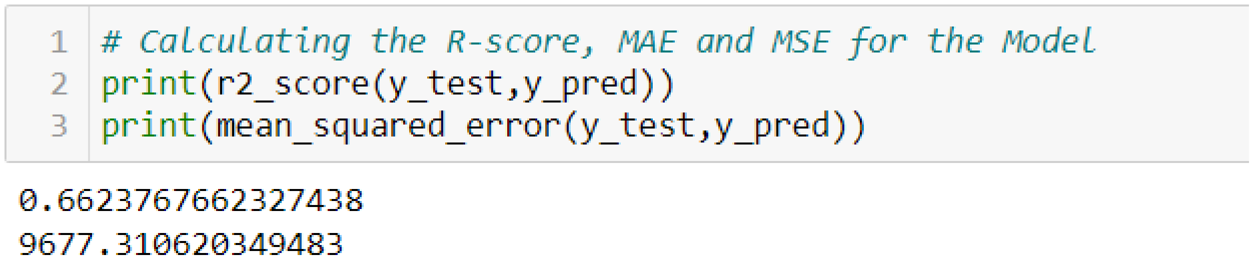


Model Evaluation

To evaluate the model's performance, it will be tested on the testing dataset that the model has not seen during training. The following evaluation metrics will be used:

Mean Squared Error (MSE): Measures the average squared difference between the predicted stock prices and the actual prices.

R-squared (R^2) Score: Indicates the proportion of the variance in the dependent variable (stock prices) that is predictable from the independent variables (features).

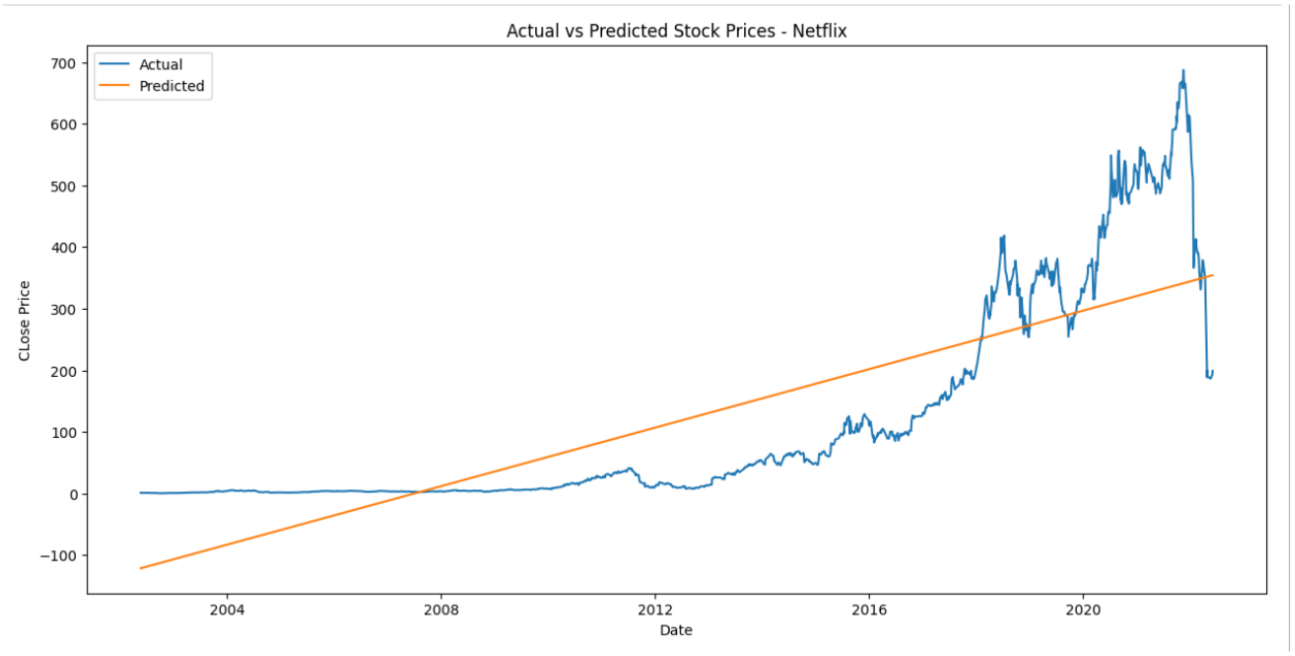


```
# Calculating the R-score, MAE and MSE for the Model
print(r2_score(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
```

```
0.6623767662327438
9677.310620349483
```

Visualization

To provide better insights and a more intuitive understanding of the results, various visualizations will be created. This will include plots of the actual vs. predicted stock prices, trends over time, and the correlation between features and stock prices.



Conclusion

The project aims to predict stock prices using a linear regression model based on historical stock price data and relevant financial indicators. By analyzing past trends and patterns, the model can make predictions about future stock prices, helping investors and traders in their decision-making process.

K-Fold Cross-Validation and Hyperparameter Tuning play crucial roles in building a robust and optimized model. K-Fold Cross-Validation allows us to assess the model's performance on multiple subsets of the data, providing a better understanding of its generalization capabilities. Hyperparameter tuning further enhances the model's performance by selecting the optimal combination of hyperparameter values.

By combining these techniques, our stock price prediction model will be better equipped to handle different market conditions and uncertainties, leading to more accurate predictions. However, as mentioned earlier, predictions should always be used in conjunction with other tools and expert advice when making investment choices.

ASD
Skills For Employment

Certificate from the Mentor

This is to certify that **[Jojangandha Saha]** has successfully completed the project titled **[Stock Price Prediction]** under my supervision during the period from February to May which is in partial fulfillment of requirements for the award of the B.Tech and submitted to Department **[Computer Science & Engineering(CSE)]** of **[Heritage Institute of Technology]**

____

Signature of the Mentor

Date: 25/07/2023

Acknowledgement

I take this opportunity to express my deep gratitude and sincerest thanks to my project mentor, **MR. Dhruba Ray** for giving the most valuable suggestion, helpful guidance and encouragement in the execution of this project work.

I would like to give a special mention to my colleagues. Last but not the least I am grateful to all the faculty members of **Academy of Skill Development** for their support.