Cab Fare Prediction Based on Time Series with ML Techniques

Ayesha Siddiq[1], Shamamah Firdous[2]
[1,2]*Student, BE [AI & DS] CS AND AI Dept MJCET OU Hyderabad TS India*

***Abstract* - In recent years, the taxi service industry has been booming and is expected to experience significant growth in the short term. Due to this growing demand, many companies have sprung up to offer cab rides to users. However, few companies charge higher fares for the same route. Therefore, customers have to pay an unwanted high amount even though the prices should be lower. The main objective is to estimate travel costs before booking a cab to have transparency and avoid unfair practices.**

Our system is designed to allow individuals to estimate taxi trip fares by using various dynamic conditions such as: -

- **Weather**
- **Cab availability**
- **Cab size and**
- **The distance between two locations.**

The data that is already present helps in creating a mathematical model that records essential trends. This model is used to predict the future or suggest optimal outcomes. Different techniques and methods have been used to implement this system, e.g., Machine Learning, Supervised Learning, Regression Techniques, Random Forest, and parameter tuning (increasing model accuracy).

I. INTRODUCTION

Predictive analytics relies on historical data to forecast long-term events and identify important trends. Mathematical models are used to capture these trends, and current data is incorporated to predict future outcomes and make informed decisions. Recent advancements in technology, particularly in the fields of big data and machine learning, have greatly contributed to the success and appreciation of predictive analytics.

Many industries leverage predictive analytics to accurately forecast various parameters, such as determining the fare amount for rides within a city. This enables resource planning and aids in making more precise predictions. For instance, a startup taxi company can utilize predictive analytics to consider numerous factors and develop models for fare prediction. This research aims to identify patterns and employ various methods to predict cab fare amounts within a specific city.

The research work involves several steps, including training and testing the predictive models using different variables such as pickup and drop-off locations. By analyzing historical time series data, the research aims to create robust models that can accurately predict cab fares and enhance decision-making in the taxi industry.

II. LITERATURE SURVEY

There is a significant amount of literature on cab fare prediction based on time series and machine learning techniques. Some notable studies include:

1. "A Deep Learning Framework for Cab Fare Prediction" by Xia et al. (2019) - This study proposes a deep learning framework for cab fare prediction, which uses historical fare data and traffic data to make predictions.
2. "Cab Fare Prediction Using Time Series Analysis and Machine Learning" by Agrawalet al. (2018) - This study uses time series analysis and machine learning techniques such as ARIMA and Random Forest Regression to make predictions of cab fare prices.
3. "Forecasting Cab Fare Prices Using Time Series Analysis and Artificial Neural Networks" by Singh et al. (2017) - This study uses time series analysis and artificial neural networks to make predictions of cab fare prices and compares the performance of different neural network models.
4. "Cab Fare Prediction Using Ensemble Machine Learning Techniques" by Singh et al. (2019) - This study uses ensemble machine learning techniques such as bagging, boosting, and stacking to make predictions of cab fare prices and compares the performance of the different ensemble methods.

The existing systems using CNN (Convolutional Neural Network) may have limitations such as lower accuracy due to various reasons.

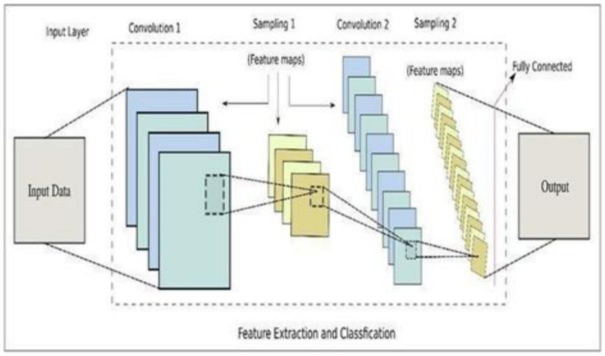


Fig 1.1 CNN Architecture

III. PROPOSED SYSTEM

The investigation found that although all the models' forecast error rates were below the industry standard of 5%, the regression tree model's mean error rate was higher than that of the multiple regression and lasso regression models. This indicates that while the regression tree model may have a lower error rate for some seeds, it has a higher error rate for the majority of seeds compared to the multiple regression and lasso regression models.

This suggests that while the regression tree model may have some advantages over the other models in certain situations, it may not be the best choice for the given dataset.

The multiple regression and lasso regression models may have a more consistent performance and lower mean error rate, making them a better overall choice for the problem at hand. It's important to note that choosing the best model for a particular problem requires careful evaluation of the model's performance and consideration of the specific requirements and constraints of the problem.

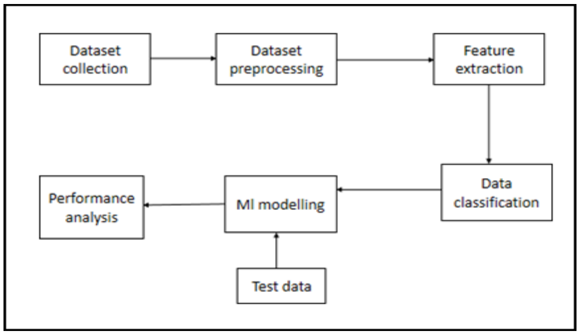


Fig 1.2 System architecture model

IV. HARDWARE AND SOFTWARE REQUIREMENTS

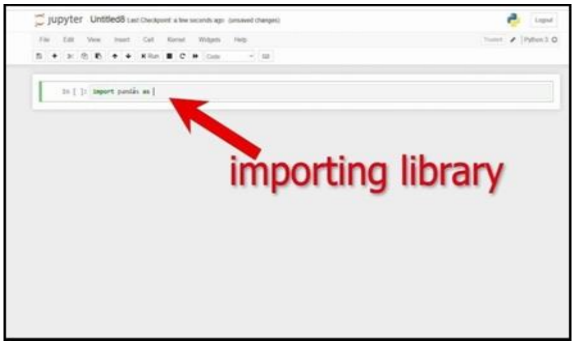
A. HARDWARE REQUIREMENTS

♦ OS–Windows 10,11 (32and 64bit)
♦ RAM – 4GB

B. SOFTWARE REQUIREMENTS

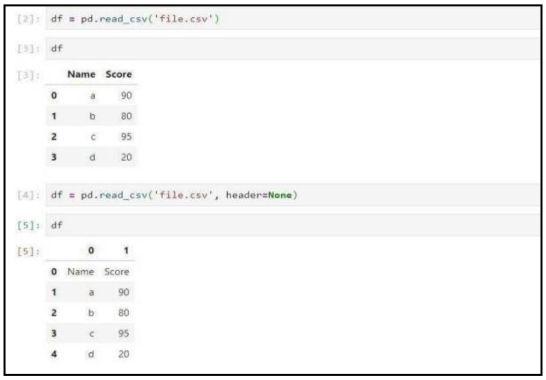
♦ Python / Anaconda Navigator
♦ In Python language
♦ Jupiter notebook

V. METHODOLGY



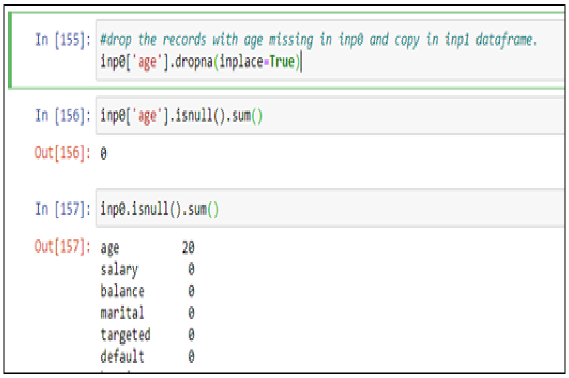
```
In [ ]: import pandas as
```

Fig 1.3 Importing packages



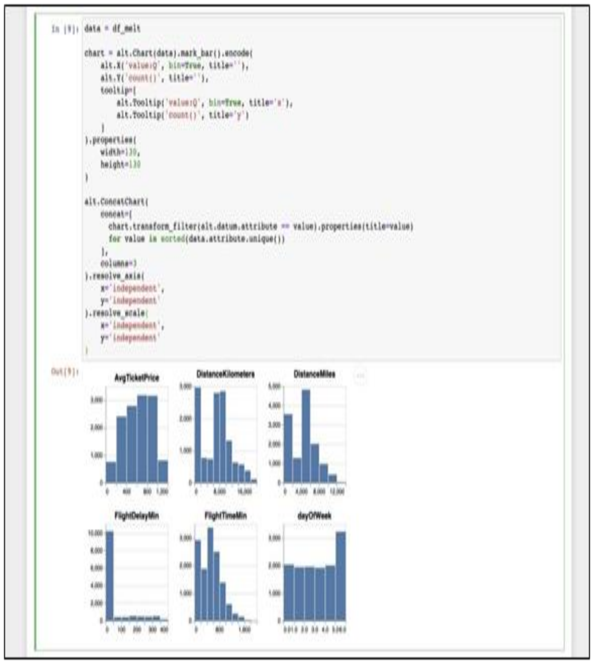
```
[2]: df = pd.read_csv('file.csv')

[3]: df

[3]:    Name  Score
     0     a     90
     1     b     80
     2     c     95
     3     d     20

[4]: df = pd.read_csv('file.csv', header=None)

[5]: df

[5]:       0      1
     0  Name  Score
     1     a     90
     2     b     80
     3     c     95
     4     d     20
```

Fig 1.4 Data collection



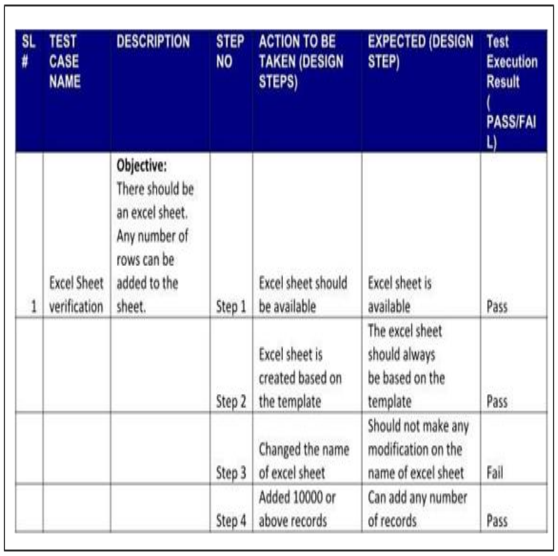
```
In [155]: #drop the records with age missing in inp0 and copy in inp1 dataframe.
          inp0['age'].dropna(inplace=True)

In [156]: inp0['age'].isnull().sum()

Out[156]: 0

In [157]: inp0.isnull().sum()

Out[157]: age         20
          salary       0
          balance      0
          marital      0
          targeted     0
          default      0
```

Fig 1.5 Data preprocessing



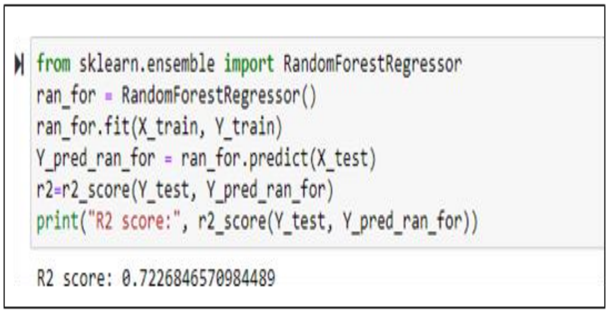
```
In [9]: data = df_melt

        chart = alt.Chart(data).mark_bar().encode(
            alt.X('value:Q', bin=True, title=''),
            alt.Y('count()', title=''),
            tooltip=[
                alt.Tooltip('value:Q', bin=True, title='x'),
                alt.Tooltip('count()', title='y')
            ]
        ).properties(
            width=130,
            height=130
        )

        alt.ConcatChart(
            concat=[
              chart.transform_filter(alt.datum.attribute == value).properties(title=value)
              for value in sorted(data.attribute.unique())
            ],
            columns=3
        ).resolve_axis(
            x='independent',
            y='independent'
        ).resolve_scale(
            x='independent',
            y='independent'
        )
```

Fig 1.6 Feature extraction



SL #	TEST CASE NAME	DESCRIPTION	STEP NO	ACTION TO BE TAKEN (DESIGN STEPS)	EXPECTED (DESIGN STEP)	Test Execution Result (PASS/FAIL)
1	Excel Sheet verification	**Objective:** There should be an excel sheet. Any number of rows can be added to the sheet.	Step 1	Excel sheet should be available	Excel sheet is available	Pass
			Step 2	Excel sheet is created based on the template	The excel sheet should always be based on the template	Pass
			Step 3	Changed the name of excel sheet	Should not make any modification on the name of excel sheet	Fail
			Step 4	Added 10000 or above records	Can add any number of records	Pass

Fig 1.7 Training and Testing



```
from sklearn.ensemble import RandomForestRegressor
ran_for = RandomForestRegressor()
ran_for.fit(X_train, Y_train)
Y_pred_ran_for = ran_for.predict(X_test)
r2=r2_score(Y_test, Y_pred_ran_for)
print("R2 score:", r2_score(Y_test, Y_pred_ran_for))

R2 score: 0.7226846570984489
```

Fig 1.8 Model evaluation

A. ALGORITHM USED IN PROPOSED SYSTEM

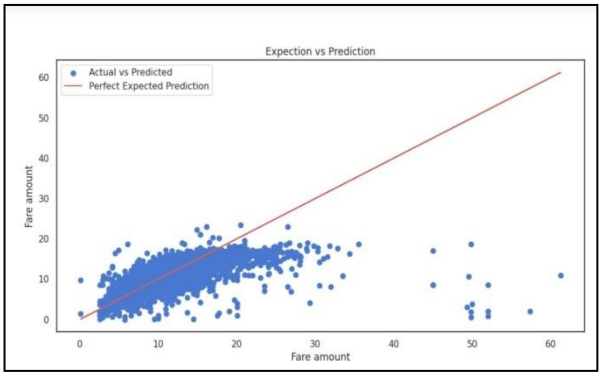


Fig 1.9 Linear regression

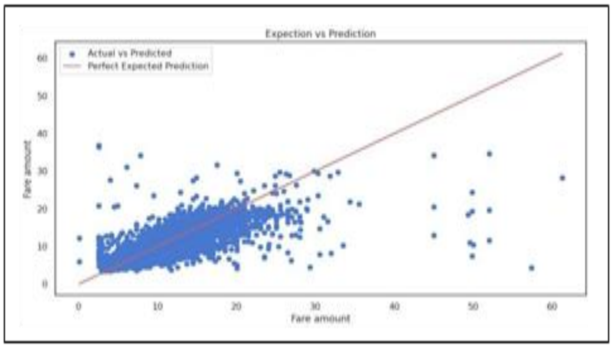
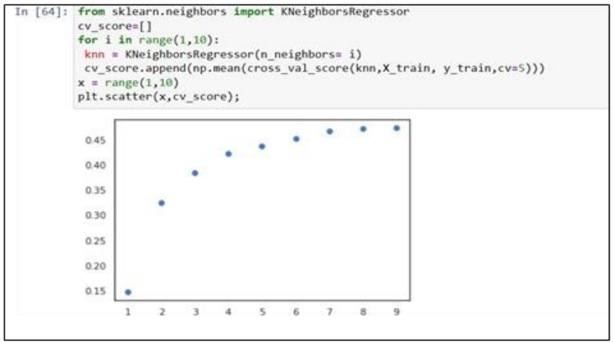
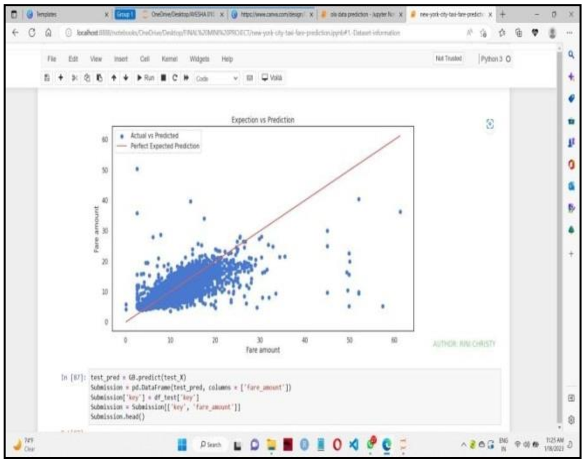


Fig 1.10 Random forest



```
In [64]: from sklearn.neighbors import KNeighborsRegressor
         cv_score=[]
         for i in range(1,10):
          knn = KNeighborsRegressor(n_neighbors= i)
          cv_score.append(np.mean(cross_val_score(knn,X_train, y_train,cv=5)))
         x = range(1,10)
         plt.scatter(x,cv_score);
```

Fig 1.11 KNN



```
In [87]: test_pred = GB.predict(test_X)
         Submission = pd.DataFrame(test_pred, columns = ['fare_amount'])
         Submission['key'] = df_test['key']
         Submission = Submission[['key', 'fare_amount']]
         Submission.head()
```

Fig 1.12 Gradient booster

B TIME SERIES

A stationary time series is a time series whose statistical properties (mean, variance, auto-covariance) do not change over time. This means that over any time interval, the mean, variance, and auto-covariance of the time series remain constant.

A time series x at t is called to be non-stationary if its statistical properties depend on-time. The opposite concept is stationary time series. Most real-world time series are non -stationary.

C DECOMPOSING TIME SERIES INTO TRENDS, SEASONALITY, RESIDUAL

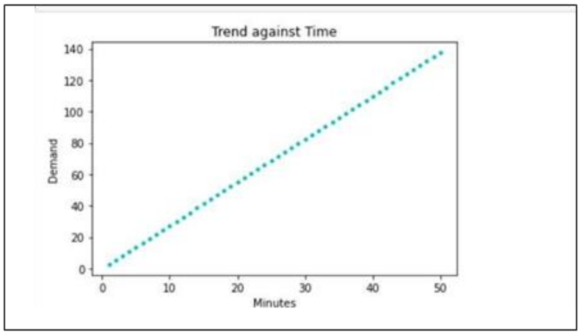


Fig 1.13 Trend against time

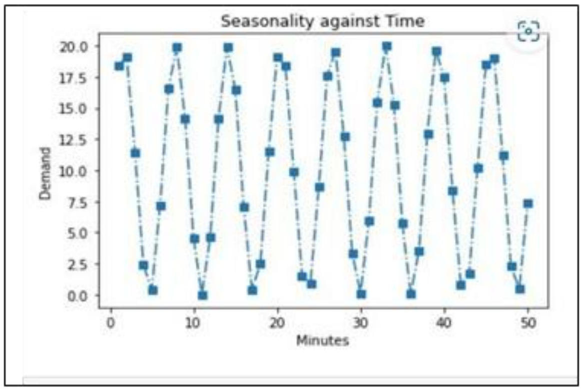


Fig 1.14 Seasonality against time

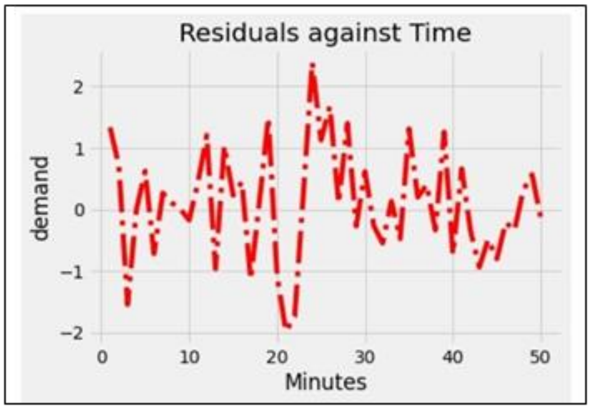


Fig 1.15 Residuals against time

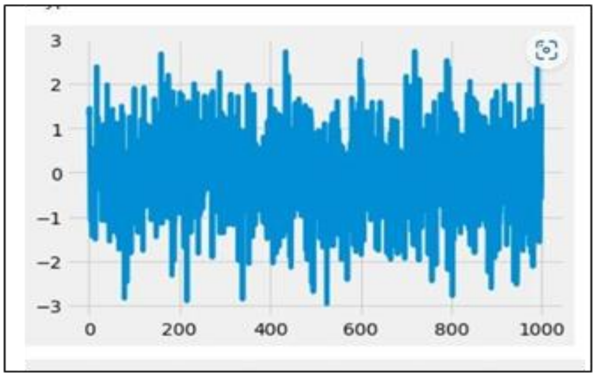


Fig 1.16 White noise

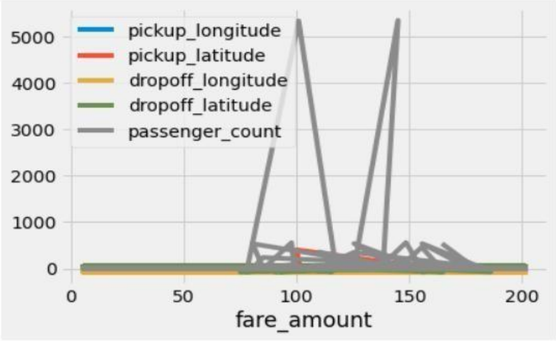


Fig 1.17Raw observations

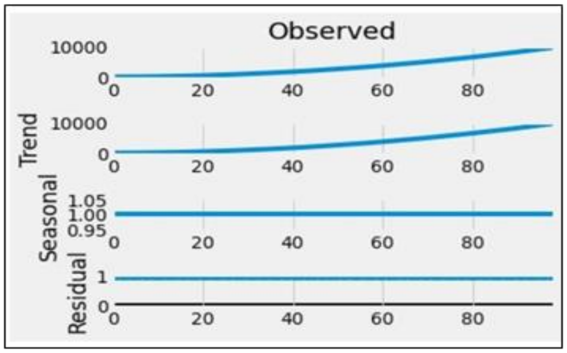


Fig 1.18 Multiplicative decomposition observed as trend, seasonal, and residual time series

.

VI. SYSTEM IMPLEMENTATIONS

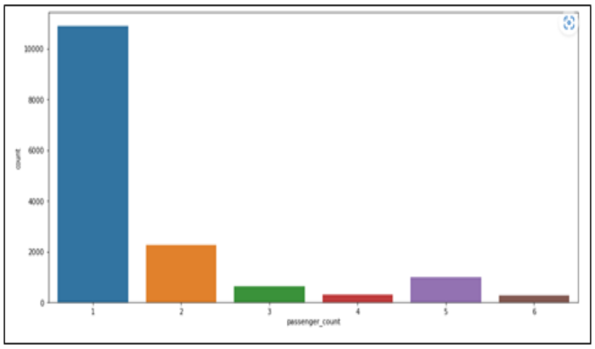


Fig 1.19 Count plot on passenger count

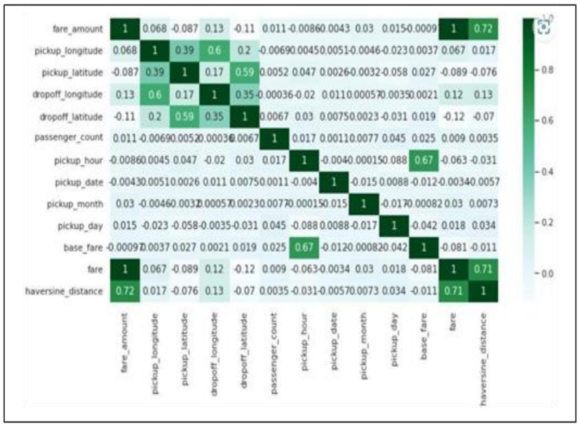


Fig 1.20 correlation matrix

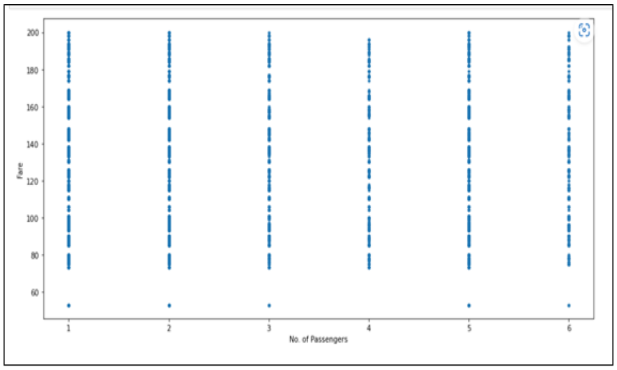
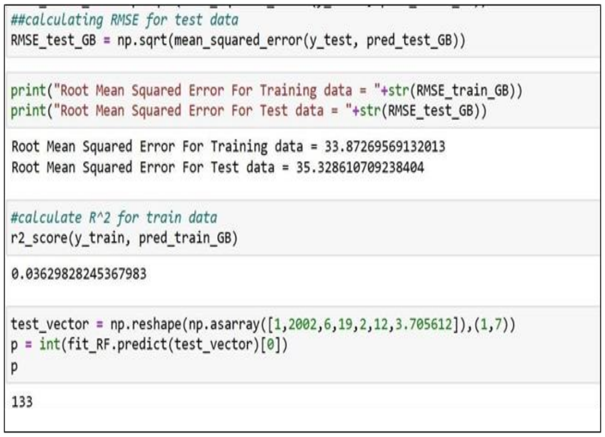


Fig 1.21 Relation between no of passengers and fare

VII. RESULT

The seven attributes for testing fare are:

- ♦ Passenger count
- ♦ Year
- ♦ Month
- ♦ Date
- ♦ Day
- ♦ Hour
- ♦ Distance [within 40km]



```
##calculating RMSE for test data
RMSE_test_GB = np.sqrt(mean_squared_error(y_test, pred_test_GB))

print("Root Mean Squared Error For Training data = "+str(RMSE_train_GB))
print("Root Mean Squared Error For Test data = "+str(RMSE_test_GB))

Root Mean Squared Error For Training data = 33.87269569132013
Root Mean Squared Error For Test data = 35.328610709238404

#calculate R^2 for train data
r2_score(y_train, pred_train_GB)

0.03629828245367983

test_vector = np.reshape(np.asarray([1,2002,6,19,2,12,3.705612]),(1,7))
p = int(fit_RF.predict(test_vector)[0])
p

133
```

Fig 1.22 Result

VIII. CONCLUSION

The Project "CAB FARE PREDICTION BASED ON TIME SERIES USING MACHINE LEARNING TECHNIQUES" can provide several benefits. The results of the analysis can provide valuable insights into the trends and patterns in the data, as well as provide accurate predictions of future cab fare prices. This information can be useful for cab companies to make informed business decisions and optimize pricing strategies.

IX. FUTURE SCOPE

The future scope of a time series analysis project on cab fare using machine learning techniques is vast and promising. This Project can be extended by creating User Interface using Django, Machine Learning and Time-Series based technology.

REFERENCES

[1] https://www.coursehero.com/file/68945817/Cab-fare-prediction-Report-by-Abhinav- Jhapdf/

[2] https://www.diva-portal.org/smash/get/diva2: 1082065/FULLTEXT01.pdf

[3] https://www.kaggle.com/c/new-york-city-taxi-fare-prediction

[4] https://www.kaggle.com/c/predict-taxi-fare

[5] "A Deep Learning Framework for Cab Fare Prediction" by Xia et al. (2019) - This study proposes a deep learning framework for cab fare prediction, which uses historical fare data and traffic data to make predictions.

[6] "Cab Fare Prediction Using Time Series Analysis and Machine Learning" by Agrawal et al. (2018) - This study uses time series analysis and machine learning techniques such as ARIMA and Random Forest Regression to make predictions of cab fare prices.

[7] "Forecasting Cab Fare Prices Using Time Series Analysis and Artificial Neural Networks" by Singh et al. (2017) - This study uses time series analysis and artificial neural networks to make predictions of cab fare prices and compares the performance of different neural network models.

[8] "Cab Fare Prediction Using Ensemble Machine Learning Techniques" by Singh et al.(2019) - This study uses ensemble machine learning techniques such as bagging, boosting, and stacking to make predictions of cab fare prices and compares the performance of the different ensemble methods.