Mentormind Hackathon

Project Report

On

"*Using Data Available from Restaurants, Predict the Only Order Delivery Time Based on certain Factors*"

Submitted By -

Data Scientists

Team Members – Kush Agrawal, Gaurav Bagade, Ameya Patil, Vikrant Rana, Haris Akram, Saurav Modak

PROJECT DETAILS AND PREREQUISITES

PROBLEM STATEMENT

We are given a data which contains information of 1000s of restaurants such as cuisines they offer, their location, ratings, votes, reviews, delivery time (which we will be predicting) from which we have to gain various insights and build a ML model to predict the delivery time.

COMPLETE PROCESS

The training dataset given has 9 columns –

1) Restaurant – ID of the restaurant
2) Location – location of restaurant
3) Cuisines – cuisines restaurants offer
4) Average_Cost – average cost per person
5) Minimum_Order
6) Rating
7) Votes
8) Reviews
9) Delivery_Time

As Cuisines column has comma separated values, we will be splitting it into 8 different columns (8 because max comma separated values gathered was 8) by splitting the cuisines column by “comma”. After we clean the data and get data which will be suitable for our ML models, we will label encode the columns – Location, 8 Cuisine variables which will be 9 of our predictor variables. The other predictor variables will be Average_Cost and Minimum_Order. Our target variable is Delivery_Time. We will be treating null values of Average_Cost with the mean of the column as it is a continuous value. After we label encode, we will be replacing those labels in our dataset.

We will start with building the model (Linear Regression) and training it with our training data and then test it using validation data using the accuracy metric "MAPE". The formula for MAPE is –

$$\frac{1}{N}\sum_{t=1}^{N}\frac{ABS(Actual_t - Forecast_t)}{Actual_t} * 100\%$$

We will start with predicting on the other given data (test data) and fit out predictions into the dataset.

Next, we will start with changing the problem statement from regression problem to classification problem. For this, we will be training multiple models and checking their accuracy and the model with highest validation accuracy will be taken. Before modeling, we would have to label encode our target variable (Delivery_Time). After the model selection is done, we will be predicting our answers for test data provided and fit our predictions into the test data.

CODES FOR ML AND DATASET

Data at initial stage –

	Restaurant	Location	Cuisines	Average_Cost	Minimum_Order	Rating	Votes	Reviews	Delivery_Time
0	ID_6321	FTI College, Law College Road, Pune	Fast Food, Rolls, Burger, Salad, Wraps	₹200	₹50	3.5	12	4	30 minutes
1	ID_2882	Sector 3, Marathalli	Ice Cream, Desserts	₹100	₹50	3.5	11	4	30 minutes
2	ID_1595	Mumbai Central	Italian, Street Food, Fast Food	₹150	₹50	3.6	99	30	65 minutes
3	ID_5929	Sector 1, Noida	Mughlai, North Indian, Chinese	₹250	₹99	3.7	176	95	30 minutes
4	ID_6123	Rmz Centennial, I Gate, Whitefield	Cafe, Beverages	₹200	₹99	3.2	521	235	65 minutes

Data after cleaning and label encoding –

Restaurant	Location	Average_Cost	Minimum_Order	Rating	Votes	Reviews	Delivery_Time	Cuisine_1	Cuisine_2	Cuisine_3	Cuisine_4	Cuisine_5
ID_6321	0	200	50	3.5	12	4	30	0	20.0	10.0	34.0	44.0
ID_2882	1	100	50	3.5	11	4	30	1	13.0	-1.0	-1.0	-1.0
ID_1595	2	150	50	3.6	99	30	65	2	16.0	0.0	-1.0	-1.0
ID_5929	3	250	99	3.7	176	95	30	3	9.0	7.0	-1.0	-1.0
ID_6123	4	200	99	3.2	521	235	65	4	6.0	-1.0	-1.0	-1.0

Cuisine_6	Cuisine_7	Cuisine_8
-1.0	-1.0	-1.0
-1.0	-1.0	-1.0
-1.0	-1.0	-1.0
-1.0	-1.0	-1.0
-1.0	-1.0	-1.0

NOTE – In the cuisine columns, null values were replaced by -1

ML Model (Linear Regression) –

```
#Importing the libraries
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

predictor = np.asanyarray(new_data[['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', 'Cuisine_6',\
                'Cuisine_7', 'Cuisine_8', 'Minimum_Order', 'Average_Cost']])
target = np.asanyarray(new_data['Delivery_Time'])

#Splitting the data into training and testing
X_train, X_test, y_train, y_test = train_test_split(predictor, target, test_size=0.3, random_state=42)

#Modeling
lr = LinearRegression()

#Training the model
lr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
#Prediction on training data
train_pred = lr.predict(X_train)
print('Mean Absolute Percentage Error on training data:',np.mean(np.abs((y_train - train_pred) / y_train)) * 100)

#Prediction on validation data
y_pred = lr.predict(X_test)
print('Mean Absolute Percentage Error on testing data:',np.mean(np.abs((y_test - y_pred) / y_test)) * 100)
```

```
Mean Absolute Percentage Error on training data: 21.903763302403163
Mean Absolute Percentage Error on testing data: 22.19167864792981
```

Preparing for Classification –

```
encode = {'Delivery_Time': {10:0, 20:1, 30:2, 45:3, 65:4, 80:5, 120:6}}
new_data = new_data.replace(encode)
new_data.head()
```

	Restaurant	Location	Average_Cost	Minimum_Order	Rating	Votes	Reviews	Delivery_Time	C
0	ID_2728	7	200.0	50	3.1	7	-	2	
1	ID_8353	6	250.0	99	4.0	498	272	3	
2	ID_6937	15	200.0	50	-	-	-	3	
3	ID_6721	7	150.0	50	NEW	-	-	2	
4	ID_8087	14	600.0	99	4.2	3782	1948	2	

We label encoded Delivery_Time column

Function for classification -

```
def classifier(model, data, X, y):
    X_train, X_test, y_train, y_test = train_test_split(data[X], data[y], test_size=0.3, random_state=42)
    model.fit(X_train,y_train)
    train_pred = model.predict(X_train)
    print(train_pred)
    train_accuracy = metrics.accuracy_score(train_pred,y_train)
    print('Accuracy on training data is: %s' % '{0:.3%}'.format(train_accuracy))
    test_pred = model.predict(X_test)
    print('\n',test_pred)
    test_accuracy = metrics.accuracy_score(test_pred, y_test)
    print('Accuracy on testing data is: %s' % '{0:.3%}'.format(test_accuracy))
```

Logistic Regression –

Logistic Regression

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', \
                 'Cuisine_6', 'Cuisine_7', 'Cuisine_8', 'Minimum_Order', 'Average_Cost']
model_lr =LogisticRegression()
classifier(model_lr, new_data,predictor_var,outcome_var)
```

```
[2 2 2 ... 2 2 2]
Accuracy on training data is: 66.941%

 [2 3 2 ... 2 2 3]
Accuracy on testing data is: 66.416%
```

Naïve Bayes –

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', 'Cuisine_6', 'Cuisine_7', 'Cuisine_8', 'Minimum_Order',
model_nb = GaussianNB()
classifier(model_nb, new_data,predictor_var,outcome_var)
```

```
[1 2 0 ... 1 1 0]
Accuracy on training data is: 3.902%

 [6 5 1 ... 1 1 5]
Accuracy on testing data is: 3.905%
```

K-Nearest Neighbors –

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', 'Cuisine_6', 'Cuisine_7', \
                 'Cuisine_8', 'Minimum_Order', 'Average_Cost']
model_knn = KNeighborsClassifier(n_neighbors=1)
classifier(model_knn, new_data,predictor_var,outcome_var)
```

```
[2 2 2 ... 2 2 2]
Accuracy on training data is: 90.547%

 [3 4 2 ... 2 2 2]
Accuracy on testing data is: 59.417%
```

Support Vector Classifier –

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', \
                 'Cuisine_6', 'Cuisine_7', 'Cuisine_8', 'Minimum_Order', 'Average_Cost']
model_svc = SVC(kernel='rbf', C=1, gamma=1)
classifier(model_svc, new_data,predictor_var,outcome_var)
```

```
[2 2 2 ... 2 2 2]
Accuracy on training data is: 91.075%

 [3 2 2 ... 2 2 2]
Accuracy on testing data is: 65.605%
```

Gradient Boosting Classifier

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', \
                 'Cuisine_6', 'Cuisine_7', 'Cuisine_8', 'Minimum_Order', 'Average_Cost']
model_gb = GradientBoostingClassifier(n_estimators=1000, learning_rate=0.1, max_depth=3, random_state=42)
classifier(model_gb, new_data,predictor_var,outcome_var)
```

```
[2 2 2 ... 2 2 2]
Accuracy on training data is: 80.296%

 [3 3 2 ... 2 2 3]
Accuracy on testing data is: 70.412%
```

Decision Tree –

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', \
                 'Cuisine_6', 'Cuisine_7', 'Cuisine_8', 'Minimum_Order', 'Average_Cost']
model_dt = DecisionTreeClassifier(criterion='gini')
classifier(model_dt, new_data,predictor_var,outcome_var)
```

```
[2 2 2 ... 2 2 2]
Accuracy on training data is: 92.363%

 [3 4 2 ... 2 2 2]
Accuracy on testing data is: 62.571%
```

Random Forest –

```
outcome_var='Delivery_Time'
predictor_var = ['Location','Cuisine_1', 'Cuisine_2', 'Cuisine_3', 'Cuisine_4', 'Cuisine_5', \
                 'Cuisine_6', 'Cuisine_7', 'Cuisine_8', 'Minimum_Order', 'Average_Cost']
model_rfc = RandomForestClassifier(n_estimators=100, random_state=42)
classifier(model_rfc, new_data,predictor_var,outcome_var)
```

```
[2 2 2 ... 2 2 2]
Accuracy on training data is: 92.350%

 [3 4 2 ... 2 2 3]
Accuracy on testing data is: 67.918%
```

Here, we saw that Gradient Boosting Classifier gives best accuracy in validation data so we will be using that for test data provided.

CONCLUSION

Here, we saw that building a linear regression model isn't the best idea but building a neural network might give us better results. Also, the given problem can be turned into classification problem but there are certain limitations such as we can't predict the actual delivery time if Delivery_Time was a continuous variable. As the values were limited, we were able to perform classification on the data.

Comparing classification model with linear regression model in our case, it seems that classification might have been better than linear regression model and a better choice as given data had discrete values for Delivery_Time but if we built a neural network model, it might have given better results and therefore be a better choice