FOOD DELIVERY TIME PREDICTION

COURSE PROJECT REPORT

18CSE398J -Machine Learning - Core Concepts with Applications

(2018 Regulation)

III Year/ VI Semester

Academic Year: 2022 -2023 (EVEN)

By

Varun Khachane (RA2011026010072)

Shresth Gupta (RA2011026010091)

Kumar Yash (RA2011026010102)

Under the guidance of

Dr. V. Vijayalakshmi

Assistant Professor

Department of Data Science and Business Systems

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMSFACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

MAY 2023

Table of Contents

1. ABSTRACT

The food delivery industry has seen a significant boom in recent years, especially with theadvent of food delivery apps. However, one of the biggest pain points for customers is the uncertainty around the delivery time. Delayed deliveries can be frustrating for customersand impact the overall experience of using food delivery services.

To address this issue, a food delivery time prediction project can be developed that utilizes machine learning algorithms to predict the delivery time accurately. This project aims to develop a model that takes into account various factors such as traffic, weather, distance, and historical delivery data to predict the delivery time accurately.

The project will involve collecting a dataset of delivery orders and their corresponding delivery times. This dataset will be used to train and test the machine learning model. Themodel will use a combination of regression and classification algorithms to predict the delivery time.

Once the model is developed, it can be integrated into food delivery apps to provide customers with accurate delivery time estimates. This can help to improve the overall customer experience and increase customer loyalty. Additionally, it can also help food delivery companies to optimize their delivery routes and improve their operational efficiency.

Overall, the food delivery time prediction project can be a valuable addition to the food delivery industry, improving the delivery experience for customers while also benefiting food delivery companies.

2. INTRODUCTION

The food delivery time prediction project aims to utilize machine learning algorithms to predict the delivery time accurately.

The project will involve the following steps:

- **Data Collection:** The first step is to collect a dataset of delivery orders and their corresponding delivery times. The dataset should include information such as the delivery address, distance from the restaurant, traffic conditions, weather conditions, and historical delivery data.

- **Data Preparation:** The collected data will need to be cleaned and preprocessed before it can be used to train the machine learning model. This will involve tasks such as removing duplicates, handling missing values, and converting categorical data to numerical data.

- **Model Training:** Once the data is prepared, the next step is to train the machine learning model. The model will use a combination of regression and classification algorithms to predict the delivery time accurately. The model will be trained using various features such as distance, traffic, weather, and historical delivery data.

- **Model Evaluation:** After the model is trained, it will be evaluated using a test dataset. The evaluation metrics will be used to determine the accuracy of the model and whether it is performing as expected.

- **Integration:** Once the model is trained and evaluated, it can be integrated into fooddelivery apps to provide customers with accurate delivery time estimates. The model will continuously learn from new data, which will improve its accuracy overtime.

The machine learning-based food delivery time prediction project has the potential to improve the overall customer experience and increase customer loyalty. Additionally, it can also help food delivery companies to optimize their delivery routes and improve theiroperational efficiency.

3. DATASET

Food delivery is a courier service in which a restaurant, store, or independent food- delivery company delivers food to a customer. An order is typically made either through a restaurant or grocer's website or mobile app, or through a food ordering company. The delivered items can include entrees, sides, drinks, desserts, or grocery items and are typically delivered in boxes or bags.

The delivery person will normally drive a car, but in bigger cities where homes and restaurants are closer together, they may use bikes or motorized scooters.

Below are all the features in the dataset:

- ID: order ID number
- Delivery_person_ID: ID number of the delivery partner
- Delivery_person_Age: Age of the delivery partner
- Delivery_person_Ratings: ratings of the delivery partner based on past deliveries
- Restaurant_latitude: The latitude of the restaurant
- Restaurant_longitude: The longitude of the restaurant
- Delivery_location_latitude: The latitude of the delivery location
- Delivery_location_longitude: The longitude of the delivery location
- Type_of_order: The type of meal ordered by the customer.
- Type_of_vehicle: The type of vehicle delivery partner rides
- Time_taken(min): The time taken by the delivery partner to complete the order

Dataset Link:

https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset

Dataset Snapshots:

ID	Delivery_p	Delivery_p	Delivery_p	Restauran	Restauran	Delivery_l	Delivery_l	Order_Dat	Time_Ord	Time_Ord	Weatherc	Road_traf	Vehicle_cc	Type_of_o	Type_of_v	multiple_c	Festival	City	Time_taken(min)
0x4607	INDORES1	37	4.9	22.745	75.8925	22.765	75.9125	#########	11:30:00	11:45:00	condition:	High	2	Snack	motorcycl	0	No	Urban	(min) 24
0xb379	BANGRES1	34	4.5	12.913	77.6832	13.043	77.8132	#########	19:45:00	19:50:00	condition:	Jam	2	Snack	scooter	1	No	Metropoli	(min) 33
0x5d6d	BANGRES1	23	4.4	12.9143	77.6784	12.9243	77.6884	#########	08:30:00	08:45:00	condition:	Low	0	Drinks	motorcycl	1	No	Urban	(min) 26
0x7a6a	COIMBRES	38	4.7	11.0037	76.9765	11.0537	77.0265	#########	18:00:00	18:10:00	condition:	Medium	0	Buffet	motorcycl	1	No	Metropoli	(min) 21
0x70a2	CHENRES1	32	4.6	12.9728	80.25	13.0128	80.29	#########	13:30:00	13:45:00	condition:	High	1	Snack	scooter	1	No	Metropoli	(min) 30
0x9bb4	HYDRES09	22	4.8	17.4317	78.4083	17.4617	78.4383	#########	21:20:00	21:30:00	condition:	Jam	0	Buffet	motorcycl	1	No	Urban	(min) 26
0x95b4	RANCHIRE	33	4.7	23.3697	85.3398	23.4797	85.4498	#########	19:15:00	19:30:00	condition:	Jam	1	Meal	scooter	1	No	Metropoli	(min) 40
0x9eb2	MYSRES15	35	4.6	12.3521	76.6067	12.4821	76.7367	#########	17:25:00	17:30:00	condition:	Medium	2	Meal	motorcycl	1	No	Metropoli	(min) 32
0x1102	HYDRES05	22	4.8	17.4338	78.3867	17.5638	78.5167	#########	20:55:00	21:05:00	condition:	Jam	0	Buffet	motorcycl	1	No	Metropoli	(min) 34
0xcdcd	DEHRES17	36	4.2	30.328	78.0461	30.398	78.1161	#########	21:55:00	22:10:00	condition:	Jam	2	Snack	motorcycl	3	No	Metropoli	(min) 46
0xd987	KOCRES16	21	4.7	10.0031	76.3076	10.0431	76.3476	#########	14:55:00	15:05:00	condition:	High	1	Meal	motorcycl	1	No	Metropoli	(min) 23
0x2784	PUNERES1	23	4.7	18.5625	73.9166	18.6525	74.0066	#########	17:30:00	17:40:00	condition:	Medium	1	Drinks	scooter	1	No	Metropoli	(min) 21
0xc8b6	LUDHRES1	34	4.3	30.8996	75.8093	30.9196	75.8293	#########	09:20:00	09:30:00	condition:	Low	0	Buffet	motorcycl	0	No	Metropoli	(min) 20
0xdb64	KNPRES14	24	4.7	26.4635	80.3729	26.5935	80.5029	#########	19:50:00	20:05:00	condition:	Jam	1	Snack	scooter	1	No	Metropoli	(min) 41
0x3af3	MUMRES1	29	4.5	19.1763	72.8367	19.2663	72.9267	#########	20:25:00	20:35:00	condition:	Jam	2	Buffet	electric_s	1	No	Metropoli	(min) 20
0x3aab	MYSRES01	35	4	12.3111	76.6549	12.3511	76.6949	#########	14:55:00	15:10:00	condition:	High	1	Meal	scooter	1	No	Metropoli	(min) 33
0x689b	PUNERES2	33	4.2	18.5927	73.7736	18.7027	73.8836	#########	20:30:00	20:40:00	condition:	Jam	2	Snack	motorcycl	1	No	Metropoli	(min) 40
0x6f67	HYDRES14	34	4.9	17.4262	78.4075	17.4962	78.4775	#########	20:40:00	20:50:00	condition:	Jam	0	Snack	motorcycl	NaN	No	Metropoli	(min) 41
0xc9cf	KOLRES15	21	4.7	22.5527	88.3529	22.5827	88.3829	#########	21:15:00	21:30:00	condition:	Jam	0	Meal	motorcycl	1	No	Urban	(min) 15
0x36b8	PUNERES1	25	4.1	18.5639	73.9154	18.6439	73.9954	#########	20:20:00	20:25:00	condition:	Jam	0	Snack	motorcycl	2	No	Metropoli	(min) 36
0xb816	CHENRES1	33	4.3	12.986	80.2181	13.116	80.3481	#########	19:30:00	19:45:00	condition:	Jam	2	Meal	scooter	1	No	Metropoli	(min) 39
0x539b	MUMRES0	25	4	19.2213	72.8624	19.2613	72.9024	#########	12:25:00	12:30:00	condition:	High	1	Buffet	motorcycl	1	No	Metropoli	(min) 34
0xa1b2	CHENRES0	29	4.5	13.0058	80.2507	13.1158	80.3607	#########	18:35:00	18:50:00	condition:	Medium	2	Meal	electric_s	1	No	Metropoli	(min) 15
0x3231	JAPRES16(	27	5	26.8496	75.8005	26.8796	75.8305	#########	20:35:00	20:40:00	condition:	Jam	0	Snack	motorcycl	0	No	Urban	(min) 18
0x8bc0	SURRES15	35	4.3	21.1605	72.7715	21.2505	72.8615	#########	23:20:00	23:30:00	condition:	Low	1	Drinks	scooter	0	No	Metropoli	(min) 38
0x2288	BANGRES0	32	4	12.9342	77.6158	13.0242	77.7058	#########	21:20:00	21:35:00	condition:	Jam	0	Buffet	motorcycl	1	No	Metropoli	(min) 47
0x3c5e	PUNERES0	23	4.8	18.5142	73.8384	18.6242	73.9484	#########	23:35:00	23:45:00	condition:	Low	2	Buffet	electric_s	0	No	Urban	(min) 12
0x3e60	COIMBRES	31	4.8	11.0225	76.9957	11.0525	77.0257	#########	22:35:00	22:50:00	condition:	Low	2	Drinks	motorcycl	1	No	Metropoli	(min) 26
0xbff	SURRES16	36	4.1	21.1604	72.7742	21.2104	72.8242	#########	22:35:00	22:40:00	condition:	Low	0	Drinks	motorcycl	1	No	Urban	(min) 22
0xd936	GOARES15	26	4.3	15.5132	73.7835	15.5632	73.8335	#########	23:25:00	23:35:00	condition:	Low	0	Buffet	motorcycl	0	No	Urban	(min) 21
0xd681	GOARES07	38	4.9	15.5613	73.7495	15.6013	73.7895	#########	13:35:00	13:40:00	condition:	High	1	Drinks	scooter	1	No	Urban	(min) 25
0x2876	RANCHIRE	32	3.5	0	0	0.11	0.11	#########	21:35:00	21:45:00	condition:	Jam	1	Snack	scooter	0	No	Urban	(min) 35
0x30c8	PUNERES1	32	4.6	18.5639	73.9154	18.6939	74.0454	#########	22:35:00	22:45:00	condition:	Low	2	Drinks	scooter	1	No	Metropoli	(min) 30
0xb843	PUNERES0	33	4.9	18.5514	73.8049	18.6214	73.8749	#########	18:55:00	19:10:00	condition:	Medium	1	Snack	motorcycl	1	No	Metropoli	(min) 22
0xb3a0	PUNERES1	20	4.7	18.5935	73.7859	18.6335	73.8259	#########	14:15:00	14:25:00	condition:	High	1	Snack	scooter	0	No	Urban	(min) 10
0x6531	SURRES08	20	4.8	21.1733	72.7927	21.1833	72.8027	#########	11:00:00	11:10:00	condition:	Low	2	Meal	scooter	1	No	Metropoli	(min) 19
0x4bda	HYDRES17	35	5	17.452	78.3859	17.472	78.4059	#########	09:45:00	09:55:00	condition:	Low	2	Snack	scooter	1	No	Urban	(min) 11
0x9d26	BANGRES1	26	4.9	12.9725	77.6082	12.9925	77.6282	#########	08:40:00	08:55:00	condition:	Low	2	Buffet	scooter	0	No	Metropoli	(min) 11
0x9b18	BANGRES1	22	4.8	12.9725	77.6082	13.0425	77.6782	#########	23:00:00	23:10:00	condition:	Low	1	Snack	motorcycl	1	No	Metropoli	(min) 28
0x5d99	CHENRES1	35	4.3	13.0642	80.2364	13.1342	80.3064	#########	17:25:00	17:30:00	condition:	Medium	1	Snack	motorcycl	1	No	Metropoli	(min) 33
0x4f0	MUMRES1	NaN	NaN	19.122	72.9085	19.202	72.9885	#########	NaN	18:35:00	condition:	Medium	1	Drinks	scooter	1	No	Metropoli	(min) 33

4. METHODS

The food delivery time prediction project methodology can be broken down into several steps, as follows:

1. **Data Collection:** The first step in the project is to collect a dataset of delivery orders and their corresponding delivery times. This dataset should include information such as the delivery address, distance from the restaurant, traffic conditions, weather conditions, and historical delivery data. The dataset can be collected from various sources such as food delivery companies, restaurant chains, and public datasources.
2. **Data Preparation:** The collected data will need to be cleaned and preprocessed before it can be used to train the machine learning model. This will involve tasks such as removing duplicates, handling missing values, and converting categorical data to numerical data.
3. **Feature Selection:** The next step is to select the most relevant features for the model. This will involve analyzing the data to determine which features have the most significant impact on the delivery time.
4. **Model Training:** Once the data is prepared and the features are selected, the next step is to train the machine learning model. The model will use a combination of regression and classification algorithms to predict the delivery time accurately. Themodel will be trained using various features such as distance, traffic, weather, and historical delivery data.
5. **Model Evaluation:** After the model is trained, it will be evaluated using a test dataset. The evaluation metrics will be used to determine the accuracy of the model and whether it is performing as expected.

Data collection involves obtaining data on delivery times, order details, traffic patterns, and other relevant factors. This data is then preprocessed, which involves cleaning, formatting, and transforming the data to prepare it for machine learning algorithms. Feature engineering involves selecting and extracting the most relevant features from the data, such as the distance between the restaurant and the delivery location, the time of day,and the order size.

Overall, the food delivery time prediction project methodology involves collecting data, preparing it, selecting relevant features, training a machine learning model, evaluating, and tuning the model, integrating it into food delivery apps, and deploying it into a production environment.

4A. Data Collection

This code reads a CSV file named "train.csv" using the pd.read_csv() function from the Pandas library in Python. The CSV file is expected to be encoded in UTF-8 format. The data from the CSV file is loaded into a Pandas DataFrame object named delivery_data.

After reading the CSV file, the head() method is called on the delivery_data DataFrame, which displays the first few rows of the data. This allows you to quickly inspect the data and get an overview of its structure.

```python
delivery_data = pd.read_csv("train.csv", encoding="utf-8")
delivery_data.head()
```

Python

ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_location_longitude	Order_Date	Time_Orderd
0x4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	75.912471	19-03-2022	11:30:00
0xb379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	77.813237	25-03-2022	19:45:00
0x5d6d	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	77.688400	19-03-2022	08:30:00
0x7a6a	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	77.026494	05-04-2022	18:00:00
0x70a2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	80.289982	26-03-2022	13:30:00

After analyze we identify the following attriubutes

```
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           45593 non-null  object
 1   Delivery_person_ID           45593 non-null  object
 2   Delivery_person_Age          45593 non-null  object
 3   Delivery_person_Ratings      45593 non-null  object
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Order_Date                   45593 non-null  object
 9   Time_Orderd                  45593 non-null  object
 10  Time_Order_picked            45593 non-null  object
 11  Weatherconditions            45593 non-null  object
 12  Road_traffic_density         45593 non-null  object
 13  Vehicle_condition            45593 non-null  int64
 14  Type_of_order                45593 non-null  object
 15  Type_of_vehicle              45593 non-null  object
 16  multiple_deliveries          45593 non-null  object
 17  Festival                     45593 non-null  object
 18  City                         45593 non-null  object
 19  Time_taken(min)              45593 non-null  object
```

4B. Data Preparation

This code snippet is using the SimpleImputer class from scikit-learn library to perform imputation on a DataFrame called delivery_data. The DataFrame has columns with missing values that need to be imputed.

The code imputes the missing values in the "Time_Ordered" column with the values from the "Time_Order_picked" column using the fillna method of the DataFrame.'

- We should handle Null values. I will use imputation beacuse not want to lose any information

> Mean imputation for numerical variables and mode imputation for categorical variables seems proper for this dataset
>
> > "Time_Orderd" NaN's will replace with rows "Time_Order_picked" since it is a date variable

```
from sklearn.impute import SimpleImputer

mean_cols = ["Delivery_person_Age", "Delivery_person_Ratings"]

mode_cols = ["Weatherconditions", "Road_traffic_density",
             "multiple_deliveries", "Festival", "City"]

mean_imp = SimpleImputer(missing_values=np.nan, strategy='mean')

for col in mean_cols:
    delivery_data[col] = mean_imp.fit_transform(delivery_data[col].to_numpy().reshape(-1,1))

mode_imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

for col in mode_cols:
    delivery_data[col] = mode_imp.fit_transform(delivery_data[col].to_numpy().reshape(-1,1))
delivery_data["Time_Orderd"] = delivery_data["Time_Orderd"].fillna(delivery_data["Time_Order_picked"])
```

```
delivery_data.isna().sum()
```

4C. Feature Selection

This code calculates the distance between a restaurant and a delivery point using their respective latitude and longitude coordinates. It uses the geopy.distance module to calculate the geodesic distance in kilometers. The calculate_dist function takes four arguments: la1 and lo1 which represent the latitude and longitude of the restaurant, and la2 and lo2 which represent the latitude and longitude of the delivery location.

```python
import geopy.distance

def calculate_dist(la1,lo1,la2,lo2):

    '''
    Calculate distance between restaurant and delivery point by using coordinates
    '''

    return geopy.distance.geodesic((abs(la1),abs(lo1)), (abs(la2),abs(lo2))).km

delivery_data["distance"] = delivery_data[["Restaurant_latitude", "Restaurant_longitude",
                            "Delivery_location_latitude", "Delivery_location_longitude"]].apply(lambda x: calculate_dist(*x), axis=1)

delivery_data = delivery_data.drop(["Restaurant_latitude", "Restaurant_longitude",
                            "Delivery_location_latitude", "Delivery_location_longitude"], axis=1)
```
[13]

The code defines a function prep_time(df) that takes a DataFrame df as input and performs various operations to calculate preparation time for orders. It converts the "Order_Date" and "Time_Orderd" columns in the DataFrame to datetime objects, extracts hour and minute information, and creates new columns for them. It also does the same for the "Time_Order_picked" column. Then, it defines an inner function calc_prep_time to calculate the preparation time for each order based on the hour and minute information.

```python
def prep_time(df):
    '''
    Convert order time and order_picked times to datetime objects and extract time information and calculate preperation time
    '''

    df["order_datetime"] = pd.to_datetime(df["Order_Date"] + ' ' + df["Time_Orderd"], dayfirst=True)
    df["picked_datetime"] = pd.to_datetime(df["Order_Date"] + ' ' + df["Time_Order_picked"], dayfirst=True)

    df = df.drop(["Order_Date", "Time_Orderd", "Time_Order_picked"], axis=1)

    df["ordered_hour"] = df["order_datetime"].apply(lambda x: x.hour)
    df["ordered_min"] = df["order_datetime"].apply(lambda x: x.minute)

    df["picked_hour"] = df["picked_datetime"].apply(lambda x: x.hour)
    df["picked_min"] = df["picked_datetime"].apply(lambda x: x.minute)

    def calc_prep_time(ord_hour, ord_min, pc_hour, pc_min):
        '''
        Calculating order preperation times. Careful about:
            - orders given before hour 24 and picked after hour 24
            - order preperation times bigger than 60 minutes
        '''

        if ord_hour == pc_hour:
            if pc_min > ord_min:
                return pc_min - ord_min
            else:
                return 0

        else:
            return 60 - ord_min + pc_min

    df["prep_min"] = df.iloc[:,-4:].apply(lambda x: calc_prep_time(*x), axis=1)

    return df

delivery_data = prep_time(delivery_data)
```

4D. Model Training

Train regression-based machine learning models and make predictions

- Train deep learning models
- Make predictions and compare results
- Apply:
- Standardization
- Grid search cross-validaton for hyperparameter tuning
- Examine feature importances

```
baseline_scores = {}

#XGBoost
params = {
         'max_depth': [3,6,10,50],
         'learning_rate': [0.005, 0.01, 0.05, 0.1],
         'n_estimators': [100, 250, 500, 1000]
         }

xgb = xgboost.XGBRegressor()
grid = GridSearchCV(estimator=xgb,
                    param_grid=params,
                    scoring='neg_mean_squared_error'
                    )
grid.fit(X_train, y_train)
print("Best parameters:", grid.best_params_)
print("Lowest RMSE: ", np.sqrt(-grid.best_score_))

baseline_scores["XGBoost"] = np.sqrt(-grid.best_score_)
```

```
#Decision Tree
params = { 'max_depth': [3, 6, 10, 50],
           'splitter': ['best', 'random'],
           'min_samples_split': [2, 10, 25, 50]
           }

dec_tree = DecisionTreeRegressor()
grid = GridSearchCV(estimator=dec_tree,
                    param_grid=params,
                    scoring='neg_mean_squared_error',
                    )
grid.fit(X_train, y_train)
print("Best parameters:", grid.best_params_)
print("Lowest RMSE: ", np.sqrt(-grid.best_score_))

baseline_scores["Decision_Tree"] = np.sqrt(-grid.best_score_)
```

```python
#Ada Boost
params = {
        'learning_rate': [0.01, 0.05, 0.1],
        'n_estimators': [100, 250, 500, 1000],
        'loss': ['linear', 'square', 'exponential']
        }

ada = AdaBoostRegressor()
grid = GridSearchCV(estimator=ada,
                    param_grid=params,
                    scoring='neg_mean_squared_error',
                    )
grid.fit(X_train, y_train)
print("Best parameters:", grid.best_params_)
print("Lowest RMSE: ", np.sqrt(-grid.best_score_))

baseline_scores["AdaBoost"] = np.sqrt(-grid.best_score_)
```

```python
#Random Forest
params = {
        'max_depth': [3, 6, 10, 50],
        'n_estimators': [100, 250, 500, 1000],
        'bootstrap': [True, False]
        }

rnd_forest = RandomForestRegressor()
grid = GridSearchCV(estimator=rnd_forest,
                    param_grid=params,
                    scoring='neg_mean_squared_error',
                    )
grid.fit(X_train, y_train)
print("Best parameters:", grid.best_params_)
print("Lowest RMSE: ", np.sqrt(-grid.best_score_))

baseline_scores["Random_Forest"] = np.sqrt(-grid.best_score_)
```

4E. Data Evaluation

RMSE (Root Mean Squared Error) is a commonly used metric to measure the accuracy of a prediction or a model's performance. It is a measure of the square root of the average of the squared differences between the predicted values and the actual values. RMSE is often used in machine learning and statistics to evaluate the performance of regression models, where the goal is to predict a continuous target variable.

Compare performances of algorithms on test sets with the best parameters by using 5-Fold Cross Validation with Hold-out method of the test set.

```
#Best Parameters for each algorithm
xgb = xgboost.XGBRegressor(learning_rate = 0.05, max_depth = 10, n_estimators = 1000)
ada_boost = AdaBoostRegressor(learning_rate = 0.1, loss = 'exponential', n_estimators = 250)
rnd_forest = RandomForestRegressor(max_depth = 10, n_estimators = 500, bootstrap=True)
dec_tree = DecisionTreeRegressor(max_depth = 10, min_samples_split = 50, splitter = 'best')

algorithms = {
            'XGBoost': xgb,
            'AdaBoost': ada_boost,
            'Random Forest': rnd_forest,
            'Decision Tree': dec_tree
            }
```

[40]

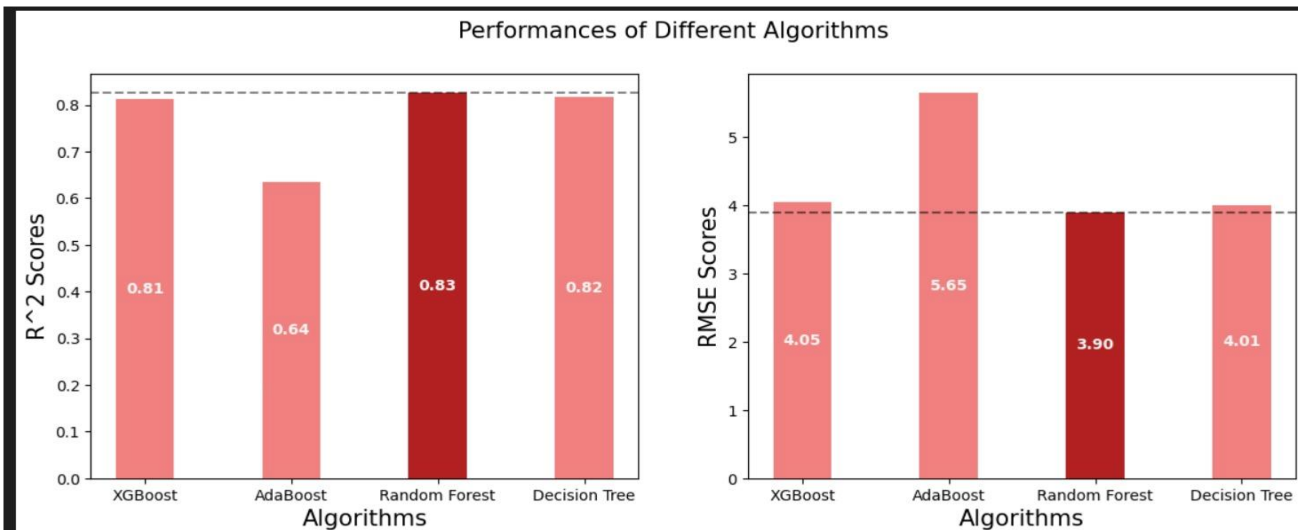
```
rmse_scores, r2_scores = {}, {}

for algorithm, model in algorithms.items():

    model.fit(X_train, y_train)
    pred = model.predict(X_test).reshape(-1,1)

    rmse_scores[algorithm] = mean_squared_error(y_test,pred)**(0.5)
    r2_scores[algorithm] = r2_score(y_test, pred)
```

[41]

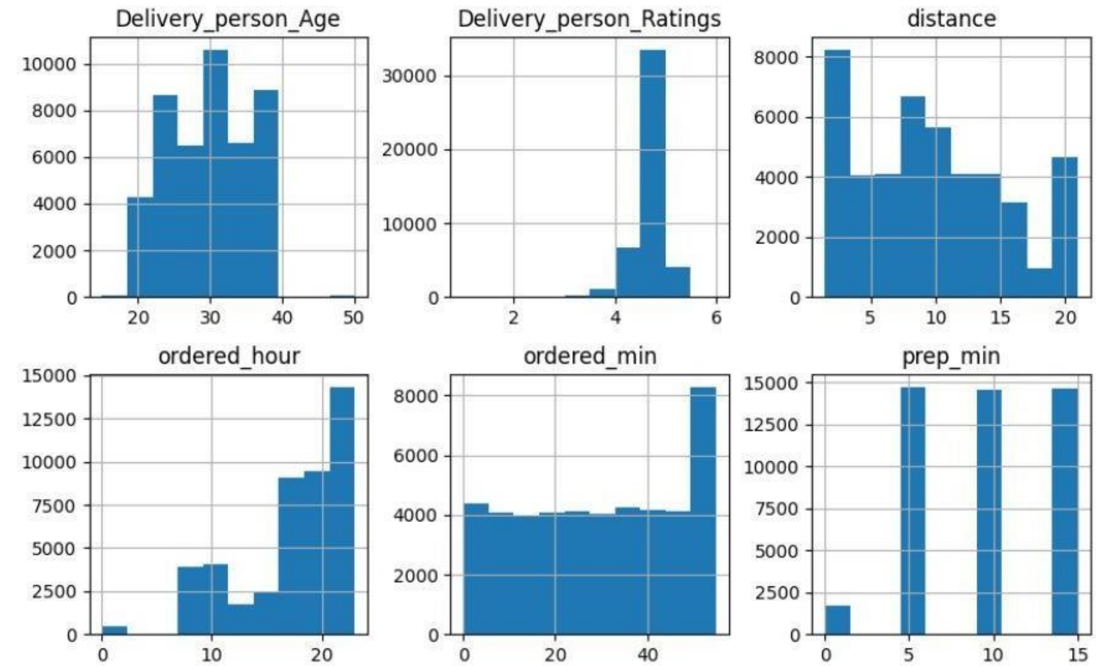


Random Forest performed best on test sets in both R^2 score and RSME score

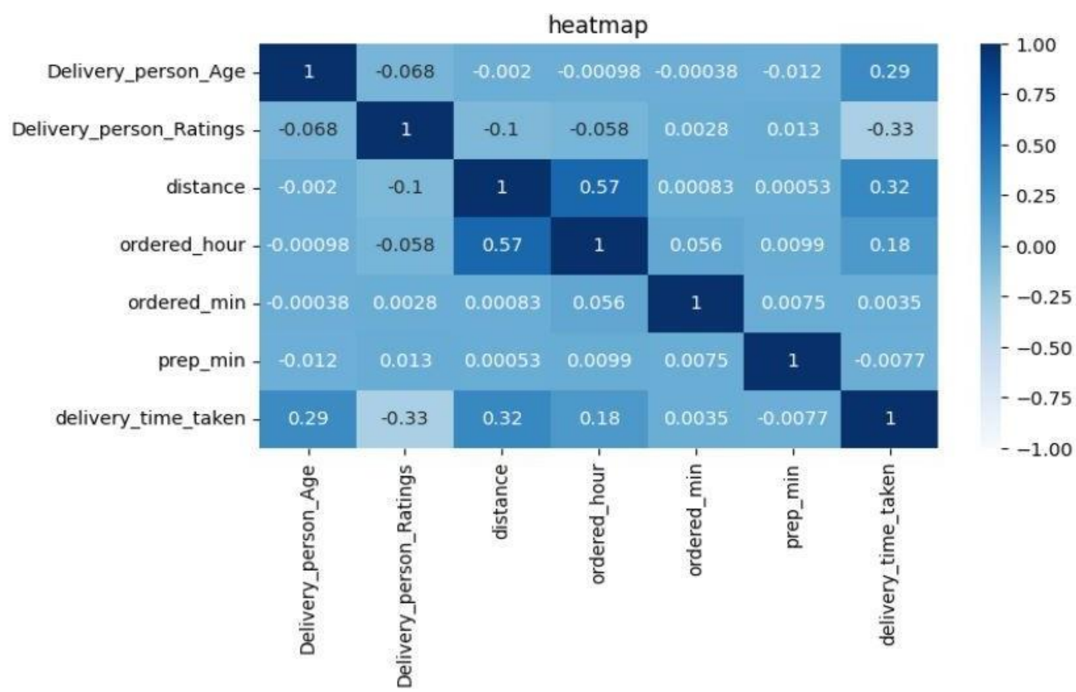
- XGBoost, Random Forest and Decision Tree performances are close to each other in regarding to R^2 Score performance
- Random Forest outperformed better in RMSE score than XGBoost and Decision Tree
- However, AdaBoost performed worst by far

+ Code + Markdown

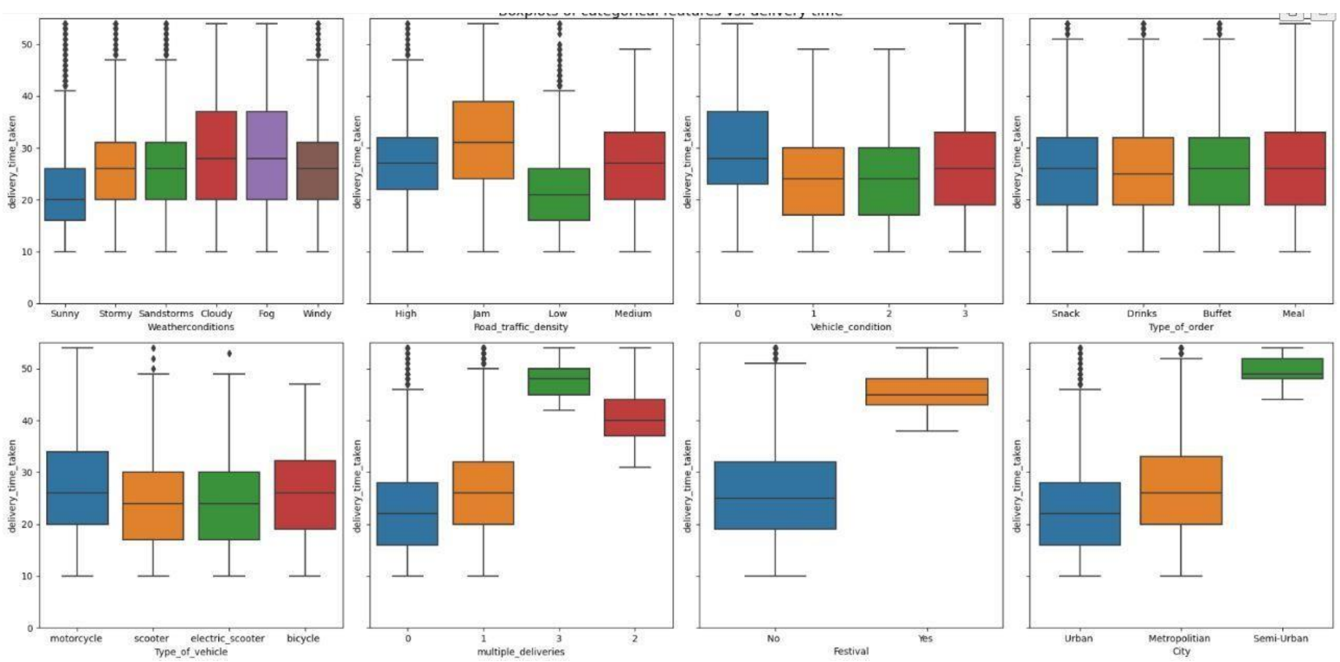
5. EXPERIMENT AND RESULTS



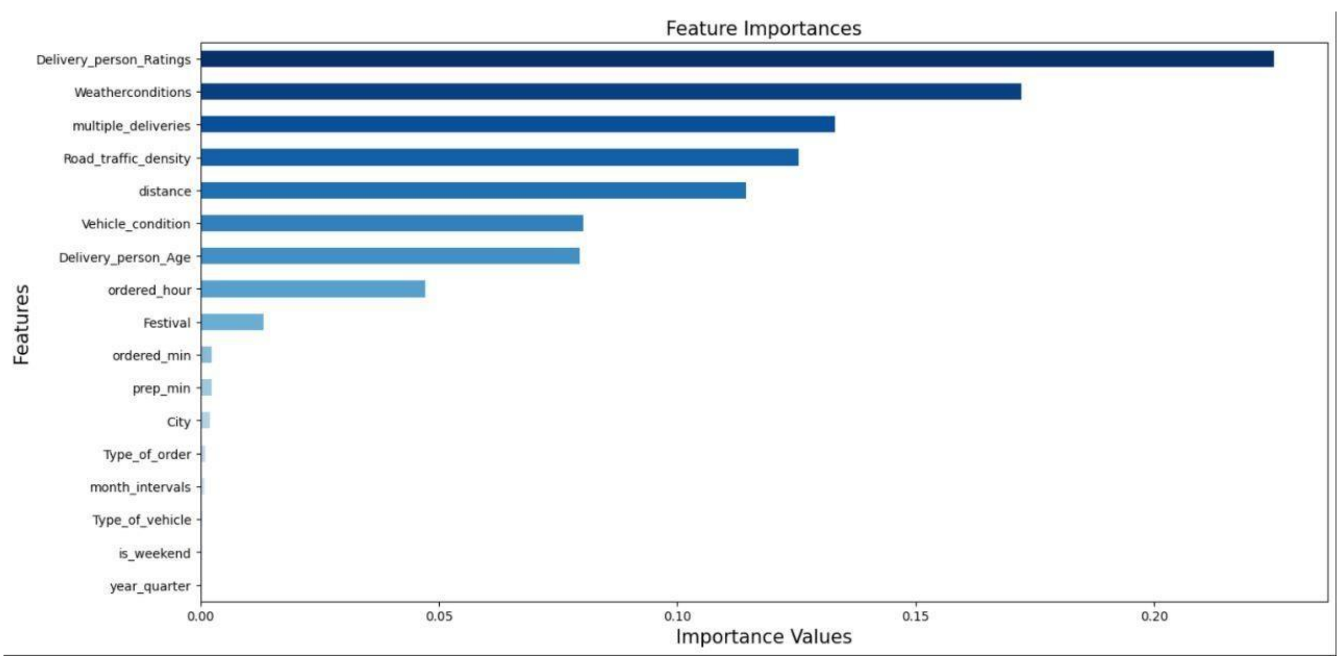
Histogram of Numerical Features in our dataset



Heatmap of the Numerical Columns



Boxplot of Categorical Columns vs delivery time



Feature Importance Graph for the given attribute

6. CONCLUSIONS AND FUTURE WORKS

The food delivery industry has grown tremendously in recent years, with the advent of food delivery services that bring restaurant-quality meals straight to people's doors. However, one common problem that plagues this industry is the issue of delivery time estimation. Customers expect their food to arrive within a reasonable timeframe, and delivery drivers need to know how long it will take to deliver an order. Accurate food delivery time prediction is, therefore, essential for improving customer satisfaction and operational efficiency.

Machine learning techniques can be used to solve this problem by predicting delivery times based on various factors such as traffic, distance, and order volume. The methodology for a food delivery time prediction project involves several steps, including data collection, data preprocessing, feature engineering, model training, and modelevaluation.

Several machine learning algorithms can be used to train a predictive model, including linear regression, decision trees, and neural networks. These algorithms use the features selected during feature engineering to make predictions about delivery times. The modelis then evaluated using metrics such as mean absolute error and mean squared error to determine its accuracy and effectiveness.

In conclusion, food delivery time prediction is an essential problem in the food delivery industry, and machine learning techniques can be used to solve it. A food delivery time prediction project involves several steps, including data collection, data preprocessing, feature engineering, model training, and model evaluation. Several machine learning algorithms can be used to train predictive models, and there are many resources available for learning machine learning concepts and techniques. Accurate food delivery time prediction can improve customer satisfaction and operational efficiency, making it a valuable application of machine learning in the food delivery industry.

7. REFERENCES

- **Research Papers:**

1. **Bhandari, R., Jain, A., & Tiwari, R. (2021).** Predicting delivery time of food orders using machine learning algorithms. Journal of Ambient Intelligence and Humanized Computing, 12(7), 6431-6443.

2. **Singhal, P., & Rastogi, A. (2020).** A survey on food delivery time prediction. International Journal of Advanced Science and Technology, 29(7), 4263-4273.

3. **Zhao, H., Cheng, Y., & Li, L. (2018).** A prediction model for food delivery time based on machine learning algorithm. In Proceedings of the 2018 3rd International Conference on Frontiers of Image Processing (ICFIP 2018) (pp. 225-230). AtlantisPress.

4. **Wang, X., Zhang, X., & Liu, S. (2019).** Delivery time prediction for food orderingand delivery platform based on time series analysis. In Proceedings of the 2019 IEEE 19th International Conference on Communication Technology (ICCT) (pp. 154-158). IEEE.

- **Books:**

1. "The Hundred-Page Machine Learning Book" by Andriy Burkov.
2. "Data Science for Business" by Foster Provost and Tom Fawcett
3. "An Introduction to Machine Learning" by Alpaydin Ethem
4. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili

These references provide insights into the different approaches and techniques used for food delivery time prediction using machine learning. They can be useful resources for understanding the project and developing the model.

- **Online Resources:**

1. **Kaggle:** Kaggle is a platform for data science competitions and provides a wealth of datasets, tutorials, and forums for machine learning practitioners.
2. **Scikit-learn:** Scikit-learn is a popular machine learning library for Python, providing a range of tools for supervised and unsupervised learning, including classification, regression, and clustering.
3. **PyTorch:** PyTorch is an open-source machine learning library for Python, providing a range of tools for deep learning, including neural networks, convolutional networks, and recurrent networks.
4. **Machine Learning Mastery:** Machine Learning Mastery is a website that provides a range of tutorials, courses, and resources for machine learning practitioners, covering topics such as data preparation, feature selection, and model evaluation.
5. **Towards Data Science:** Towards Data Science is a platform for sharing knowledge and ideas in data science, providing a range of articles and tutorials on machine learning, deep learning, and data analysis.

These websites provide a range of resources for machine learning practitioners, from tutorials and examples to libraries and datasets.

Food Delivery Time Prediction Services

1. DoorDash: https://www.doordash.com/
2. Postmates: https://postmates.com/
3. Zomato: https://www.zomato.com/
4. Swiggy: https://www.swiggy.com/
5. Deliveroo: https://deliveroo.co.uk/
6. Talabat: https://www.talabat.com/

These websites and platforms can provide examples of how food delivery time predictionis implemented in real-world scenarios. Studying these websites can also help identify best practices for predicting delivery times accurately and efficiently.