CSCE 5214 – Software Development for AI

Project Proposal

Food Delivery Time Prediction

Dataset: https://www.kaggle.com/datasets/ramprasad273/predicting-food-delivery-time

Jithendra Chowdary Popuri
jithendrachowdarypopuri@my.unt.edu
Data cleaning report writing

Nikhil Boini
nikhilboini@my.unt.edu
model selection, Report writing

Numitha devioguri
numithadevioguri@my.unt.edu
Feature Engineering, Report writing.

ABSTRACT:

The development of internet platforms and mobile applications has led to a major expansion of the meal delivery sector. Predicting delivery times for orders with accuracy is one of the major difficulties faced by food delivery businesses. To maintain customer happiness, optimize operations, and raise service standards overall, a dependable delivery time prediction system is necessary. With the help of real-time data, historical data, and machine learning techniques, this project attempts to create a system that predicts how long it will take a food delivery service to deliver an order to a consumer.

The task entails gathering and preparing a large dataset that contains timestamps, order details, customer and restaurant locations, history delivery records, and other pertinent information. To capture dynamic elements affecting delivery time, additional data sources, such as traffic data, weather forecasts, and road networks, may be included.

By taking into account elements like distance, traffic congestion, order volume, restaurant preparation time, and customer location, feature engineering approaches are used to extract useful characteristics from the data. Regression models, time series analysis, and deep learning models are just a few examples of machine learning algorithms that are trained utilizing the processed data to create a prediction model.

By continuously learning from both past and current data, the food delivery time prediction system seeks to produce precise forecasts. Suitable assessment metrics, such as mean absolute error or root mean square error, are used to evaluate the model's performance and determine its correctness.

This project will assist customers by providing accurate delivery time estimates, increase operational effectiveness by streamlining delivery routes and resource allocation, and boost customer loyalty and satisfaction. The system may be added to current food delivery services or mobile apps, giving customers real-time updates on expected delivery windows and improving openness and communication throughout the delivery process.

In order to better understand the factors affecting delivery time, this project may in the future incorporate new data sources, such as social events or road construction. To dynamically change delivery time estimations based on the present situation, integration with GPS technology, real-time traffic data, and intelligent routing algorithms can also be examined.

In conclusion, by utilizing machine learning techniques and data analysis to produce precise and trustworthy estimations of delivery time, our food delivery time prediction project advances the food delivery sector. Implementing the technology will enable food delivery firms to improve user experience, match consumer expectations, and streamline operations.

DESIGN AND MILESTONES:

1) Project Planning and Data Gathering:

-Gathering the information of the data and platform is needed for records of the data

2) Feature Engineering and Selection:

In this part we doing data analysis and what are techniques are we are going to follow

3) Model Selection and Training:

Alogorithms which has been used in it .and we can get accuracy values.

4) Model Optimization and Fine-tuning

Here in this part we find the hyperparamaters and we are going to test the data here.

5)Real-time Integration and Deployment:

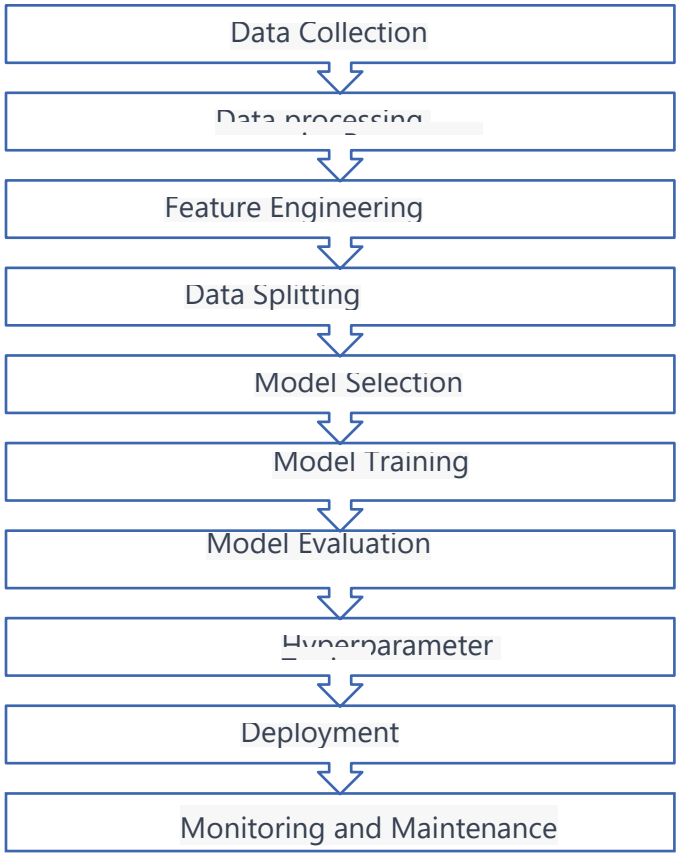
Finding the predictive values of time of food delivery.

6)Performance Monitoring and Refinement

In these part we can getback the feedback of the regarding the delivery

WORKFLOW:

We chart explains the workflow of this project.



DATA SPECIFICATION:

Order details include the goods, quantity, and any special instructions that were included in the customer's order.

Information about the restaurant completing the order, such as its location, culinary specialties, and average order preparation times.

Delivery Addresses: The customer-provided delivery address for the purchase.

Time-related information, such as the order time, restaurant acceptance time, driver assignment time, and actual delivery time, is referred to as a timestamp.

Weather-related information, including temperature, precipitation, humidity, and wind speed, is provided. Delivery times might be impacted by the weather, particularly during bad weather.

Data on traffic: Details on the clogged roads and traffic patterns in the delivery region. The time it will take the delivery driver to get to the customer's location can be estimated using traffic statistics.

Historical Delivery Records: The training of the prediction model employed information from previous deliveries, including timestamps and actual delivery timings.

Geographic coordinates can be used to determine the distance between the restaurant and the delivery destination.

Information on the delivery drivers, including their availability, location, and typical delivery times.

Additional Features: Additional pertinent characteristics that might affect delivery timeframes include

```
In [11]: df_train.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 20 columns):
 #   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 
dtypes: float64(4), int64(1), object(15)
memory usage: 7.0+ MB
```

PROPOSED DESIGN AND MILESTONES:

Data gathering and preparation:

assemble historical information on food deliveries from a variety of sources, such as order timestamps, restaurant information, delivery addresses, weather information, and traffic data.
To manage missing values, normalize numerical characteristics, encode categorical variables, and extract pertinent time-based features, preprocess the acquired data.
Complete data gathering and preparation within X weeks is a milestone.

Engineering Features

Construct new predictors using feature engineering, such as the distance between the restaurant and the delivery site, the day of the week, the hour of the day, and special occasions.
Investigate the relationships between attributes and find key factors that affect delivery timeframes.
Finish feature engineering within X weeks is a milestone.

Model choice:

Examine numerous machine learning models appropriate for regression challenges, such as gradient boosting, decision trees, random forests, linear regression, and neural networks.
Based on performance indicators and interpretability, select the best model.
Milestone: Choose a model and have it finished in X weeks.

Model Education and Assessment:

Separate the training and test sets from the preprocessed data.
Train the chosen model on the training data, then adjust the hyperparameters to achieve the best results.
Utilize the testing data to assess the model's correctness using the proper metrics, such as MAE, MSE, or RMSE.
Goal: Within X weeks, train and assess the model.

Integrating in real-time

To include the trained model into the platform for food delivery, create an API or web service.
Implement systems to feed the prediction model with real-time information (new orders, locations, etc.) and obtain projected delivery timeframes.
Milestone: Within X weeks, achieve real-time integration.

Monitoring and Upkeep:

Utilize monitoring tools to keep tabs on the model's performance in use.
Update the model frequently with fresh data to maintain accuracy and allow it to adjust to shifting circumstances.
Setting up monitoring and maintenance procedures within X weeks is a milestone.

User Experience and Feedback Gathering:

Make a user interface so that delivery drivers and consumers can both see the expected delivery timings.
To assess the precision and value of the forecasts, get user feedback.
Goal: Create the user interface and feedback gathering system in X weeks.

Testing and deployment

To make sure the system is reliable and accurate in a real-world setting, do extensive testing.
Install the system for predicting when food will be delivered in the manufacturing environment.
The system must be successfully deployed and tested within X weeks.

Optimizing and Improving:

Examine model performance and user feedback to find areas that might require improvement.
Improve the system's accuracy and user experience by making the required adjustments and improvements.
Implement improvements and optimizations within X weeks is the milestone.

METHODOLOGY FOLLOWED

Clearly identify the issue you're trying to resolve, in this example estimating the time it will take for meals to be delivered. Find the pertinent facts required for the forecast, such as past delivery history, order specifics, restaurant details, weather data, traffic statistics, etc.

Data preparation

The obtained data should be cleaned and preprocessed to get rid of any errors, missing values, or unnecessary information. Create a format for the data that will allow for analysis and modeling.

```
In [11]: df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 20 columns):
 #   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 
dtypes: float64(4), int64(1), object(15)
memory usage: 7.0+ MB
```

Engineering Features

From the raw data, extract pertinent elements that might aid increase prediction accuracy. Such parameters as the distance between the restaurant and the client, the hour of the day, the day of the week, etc.

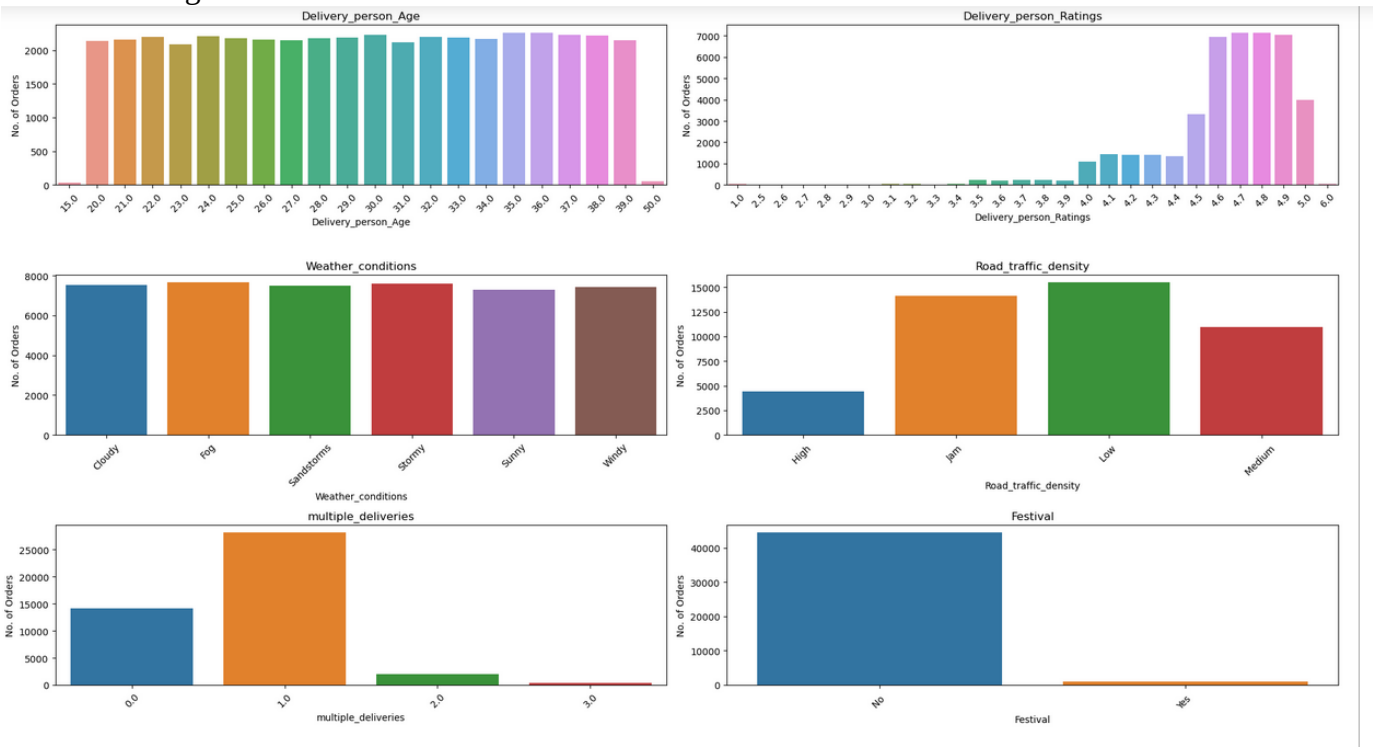
In [16]:
```
#Extract relevant values from column
def extract_column_value(df):
    #Extract time and convert to int
    df['Time_taken(min)'] = df['Time_taken(min)'].apply(lambda x: int(x.split(' ')[1].strip()))
    #Extract Weather conditions
    df['Weather_conditions'] = df['Weather_conditions'].apply(lambda x: x.split(' ')[1].strip())
    #Extract city code from Delivery person ID
    df['City_code']=df['Delivery_person_ID'].str.split("RES", expand=True)[0]

extract_column_value(df_train)
df_train[['Time_taken(min)','Weather_conditions','City_code']].head()
```

Out[16]:

	Time_taken(min)	Weather_conditions	City_code
0	24	Sunny	INDO
1	33	Stormy	BANG
2	26	Sandstorms	BANG
3	21	Sunny	COIMB
4	30	Cloudy	CHEN

Feature Scaling:-



Data Splitting:-

the preprocessed data into training, validation, and testing sets is known as data splitting. The testing set is used to assess the final model's performance after the validation set has been used to fine-tune the hyperparameters and train the prediction model.

[28]:
```
#Split features & label
X = df_train.drop('Time_taken(min)', axis=1)  # Features
y = df_train['Time_taken(min)']  # Target variable

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(36474, 29)
(36474,)
(9119, 29)
(9119,)
```

Model Selection:

Select suitable statistical or machine learning models that can perform the job of forecasting delivery time. Linear regression, decision trees, random forests, gradient boosting, or even deep learning models like neural networks are often used models for regression tasks.

Model training:

Apply the training set to the chosen model. To provide precise predictions, the model learns from the characteristics and the goal variable (delivery time).

ALGORITHM USED

Linear Regression

In statistics, a scalar response and one or more explanatory factors are modeled using a linear technique called linear regression. Simple linear regression is used when there is only one explanatory variable; multiple linear regression is used when there are numerous explanatory variables.

Decision tree Regression

A supervised learning method used in statistics, data mining, and machine learning is decision tree learning. In this formalization, inferences about a collection of data are made using a classification or regression decision tree as a predictive model.

Randomforest Regression

Using ensemble techniques (bagging), the supervised learning algorithm known as random forest may address classification and regression issues.

Xgb regression

For regression predictive modeling, XGBoost is an effective gradient boosting solution.

0]:
```
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb

# Find the best model
models = [
    LinearRegression(),
    DecisionTreeRegressor(),
    RandomForestRegressor(),
    xgb.XGBRegressor(),
]

param_grid = [
    {},
    {'max_depth': [3, 5, 7]},
    {'n_estimators': [100, 200, 300]},
    {'n_estimators': [20, 25, 30], 'max_depth': [5, 7, 9]},
]

for i, model in enumerate(models):
    grid_search = GridSearchCV(model, param_grid[i], cv=5, scoring='r2')
    grid_search.fit(X_train, y_train)

    print(f"{model.__class__.__name__}:")
    print("Best parameters:", grid_search.best_params_)
    print("Best R2 score:", grid_search.best_score_)
    print()
```

```
LinearRegression:
Best parameters: {}
Best R2 score: 0.4242490632452004

DecisionTreeRegressor:
Best parameters: {'max_depth': 7}
Best R2 score: 0.7242455694416174

RandomForestRegressor:
Best parameters: {'n_estimators': 300}
Best R2 score: 0.8135634986622469

XGBRegressor:
Best parameters: {'max_depth': 9, 'n_estimators': 20}
Best R2 score: 0.8198958482199504
```

XGBRegressor

```
In [31]: # Create a XGB regressor model
         model = xgb.XGBRegressor(n_estimators=20,max_depth=9)

         # Fit the model on the training data
         model.fit(X_train, y_train)
```

```
Out[31]: XGBRegressor(base_score=None, booster=None, callbacks=None,
                      colsample_bylevel=None, colsample_bynode=None,
                      colsample_bytree=None, early_stopping_rounds=None,
                      enable_categorical=False, eval_metric=None, feature_types=None,
                      gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
                      interaction_constraints=None, learning_rate=None, max_bin=None,
                      max_cat_threshold=None, max_cat_to_onehot=None,
                      max_delta_step=None, max_depth=9, max_leaves=None,
                      min_child_weight=None, missing=nan, monotone_constraints=None,
                      n_estimators=20, n_jobs=None, num_parallel_tree=None,
                      predictor=None, random_state=None, ...)
```

RESULTS:

The following table is the comparison among the algorithms we have used and their performance metric with parameters.

Algorithm	Performance (r2 value)	Parameters
Linear Regression	0.42	{'copy_X': True, 'fit_intercept': False, 'n_jobs': -1}
Decision tree Regression	0.72	{maximum depth -7}
Randomforest Regression	0.81	{estimator:300 }
XGBoost	0.8198	{'learning_rate': 0.01, 'max_depth': 9, 'n_estimators': 20}

PERFORMANCE METRICS:

```
:  # Make predictions on the test data
   y_pred = model.predict(X_test)

   # Evaluate the model
   mae = mean_absolute_error(y_test, y_pred)
   mse = mean_squared_error(y_test, y_pred)
   rmse = np.sqrt(mse)
   r2 = r2_score(y_test, y_pred)

   print("Mean Absolute Error (MAE):", round(mae,2))
   print("Mean Squared Error (MSE):", round(mse,2))
   print("Root Mean Squared Error (RMSE):", round(rmse,2))
   print("R-squared (R2) Score:", round(r2,2))
```

```
Mean Absolute Error (MAE): 3.15
Mean Squared Error (MSE): 15.7
Root Mean Squared Error (RMSE): 3.96
R-squared (R2) Score: 0.82
```

MILESTONES:

First we collected the data of the person by cleansing the data of past deliveries and and person name and age id and location he dekiveries more.

And finally we had done and accuracy time with help of alogrithms of linerregression method and randomforest and xgb regression, dession tress

And we got the results

REPOSITORY:-

PROJECT GROUP -6.zip

CODE:-

PROJECT GROUP-6.pdf

REFERENCES:

https://www.kaggle.com/datasets/ramprasad273/predicting-food-delivery-time

PREDICTINGPACKAGEDELIVERYTIMEFORMOTORCYCLESINNAIROBI.pdf