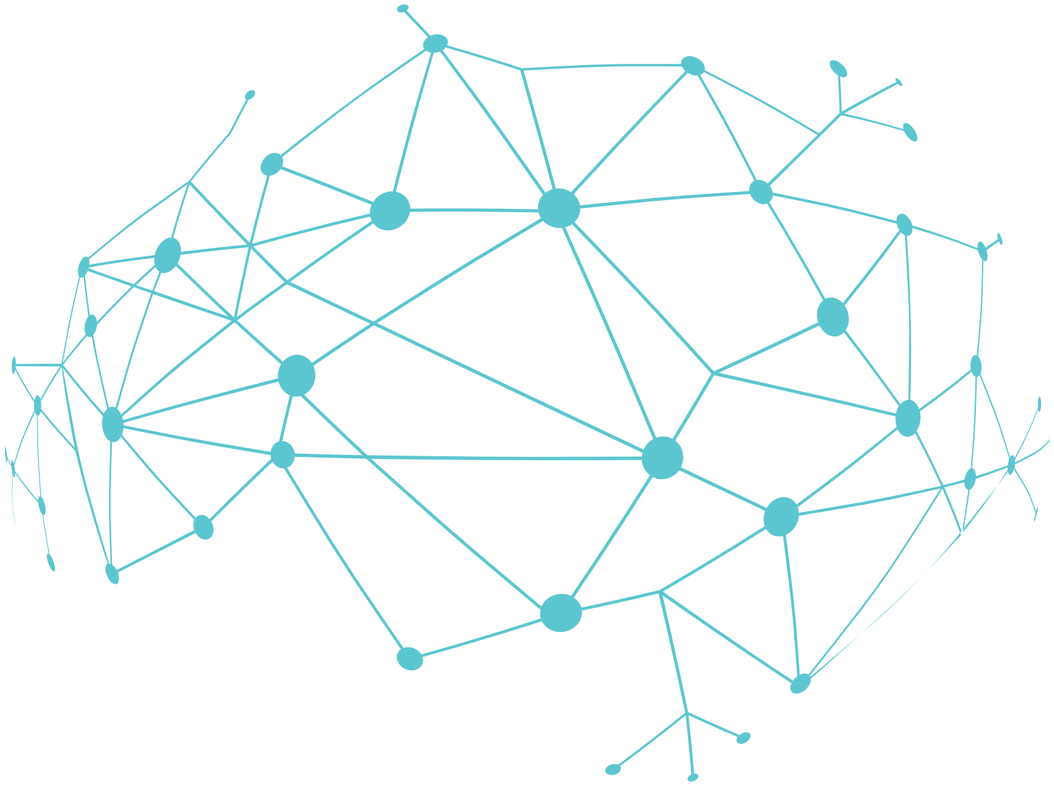
Institute of Technology of Cambodia
Deparment of Information and Communication Engineering

AI PROJECT



Topic: Public Transportation

By Group 3

2023 - 2024

Team Member

KONG VONGPISITH

ROTHA DAPRAVITH

YORNG TONGHY

Table of Content

1. Introduction

The **Public Transportation** Dataset includes key information such as transit times, GPS data, and passenger details. It's used to improve bus and train routes, predict travel times, and make better transport policies.

Objective

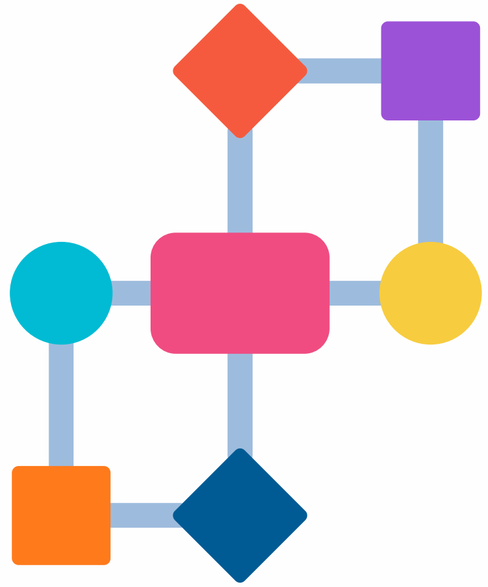
The objective of the project is to use machine learning includes:

- Improve public transportation networks.
- Estimate passenger loads,
- Predict duration trips.
- Decreasing congestion roads.

2. Machine Learning Algorithms

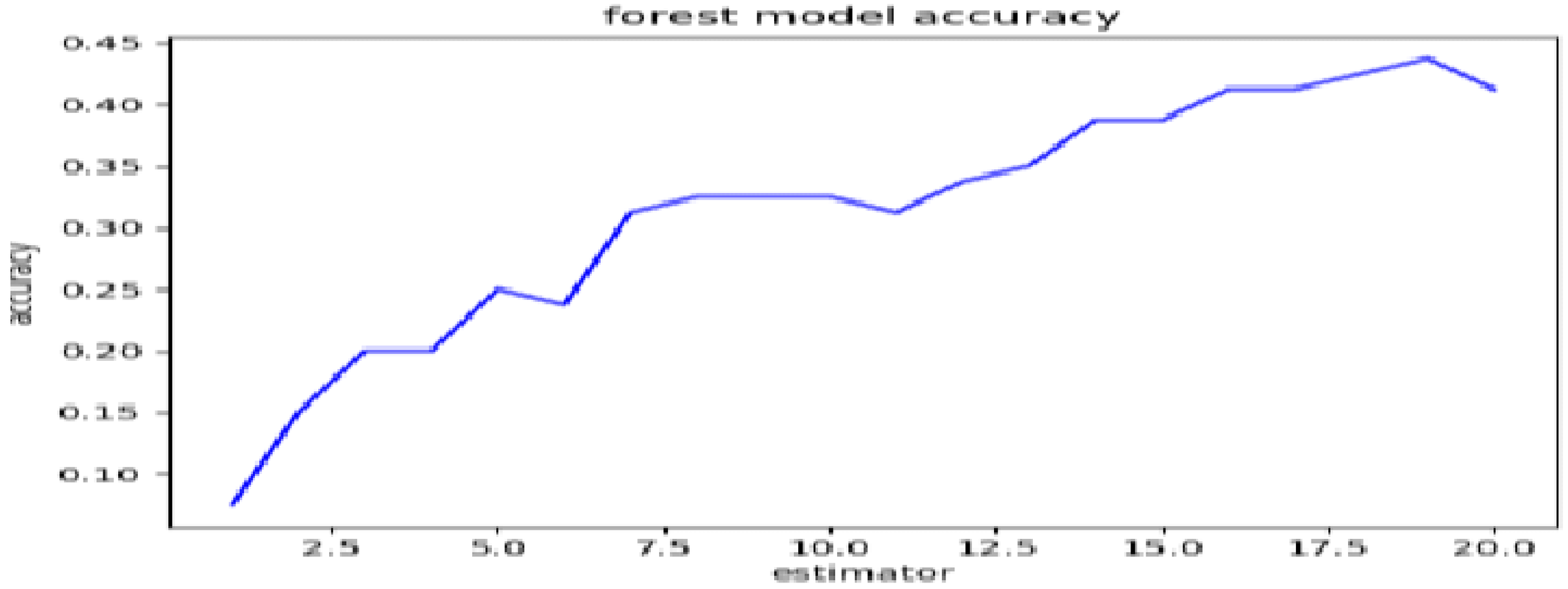
In this project, we use 3 different algorithms to train the model are:

- Random Forest.
- Linear Regression.
- Decision Tree.



2.1. Random Forest Algorithm

Random Forest is a supervised machine-learning algorithm made up of decision trees. It is used for both classification and regression problems.



2.1. Random Forest Algorithm

How to use Random Forest for train model ?

The purpose of using Random Forest in training models for public transportation prediction through machine learning includes:

- **Handing complexity dataset**
- **Importance of Features**
- **High Accuracy**
- **Overfitting Reduction**

2.1. Random Forest Algorithm

How Random Forest work?

The Working process can be explained in the below steps and diagram:

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

2.1. Random Forest Algorithm

Split dataset

```
1  from sklearn.model_selection import train_test_split
2  from sklearn import preprocessing
3
4  scaler = preprocessing.StandardScaler()
5
6  predictors = df.drop("price", axis=1)
7  df["price"] = df["price"].astype(float)
8  price = df["price"]
9
10 X_train,X_test,Y_train,Y_test = train_test_split(predictors,price,test_size=0.30,random_state=0)
```

2.1. Random Forest Algorithm

Implement algorithm code

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Assuming the best_random_state has been determined in a previous step of your analysis
best_random_state = 100

# Initialize the RandomForestClassifier with the best found random state
rf = RandomForestClassifier(random_state=best_random_state, n_jobs=-1, max_depth=7,
                            min_samples_leaf=1, min_samples_split=5,
                            n_estimators=50, oob_score=True)

# Train the model
rf.fit(X_train, y_train)

# Predict on the test set
y_pred = rf.predict(X_test)

# Calculate the accuracy and classification report
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Print the accuracy as a percentage with 2 decimal places
print("Accuracy: {:.2f}%".format(accuracy * 100))
print("Classification Report:\n", report)

```

Train Model

```
Accuracy: 92.02%
Classification Report:
                precision    recall  f1-score   support

           0       0.97      0.89      0.93     34736
           1       1.00      1.00      1.00    106241
           2       0.79      0.65      0.72      6300
           3       0.70      0.97      0.81     30726
           4       0.92      0.68      0.78     30704

    accuracy                           0.92    208707
   macro avg       0.88      0.84      0.85    208707
weighted avg       0.93      0.92      0.92    208707
```

Accuracy score

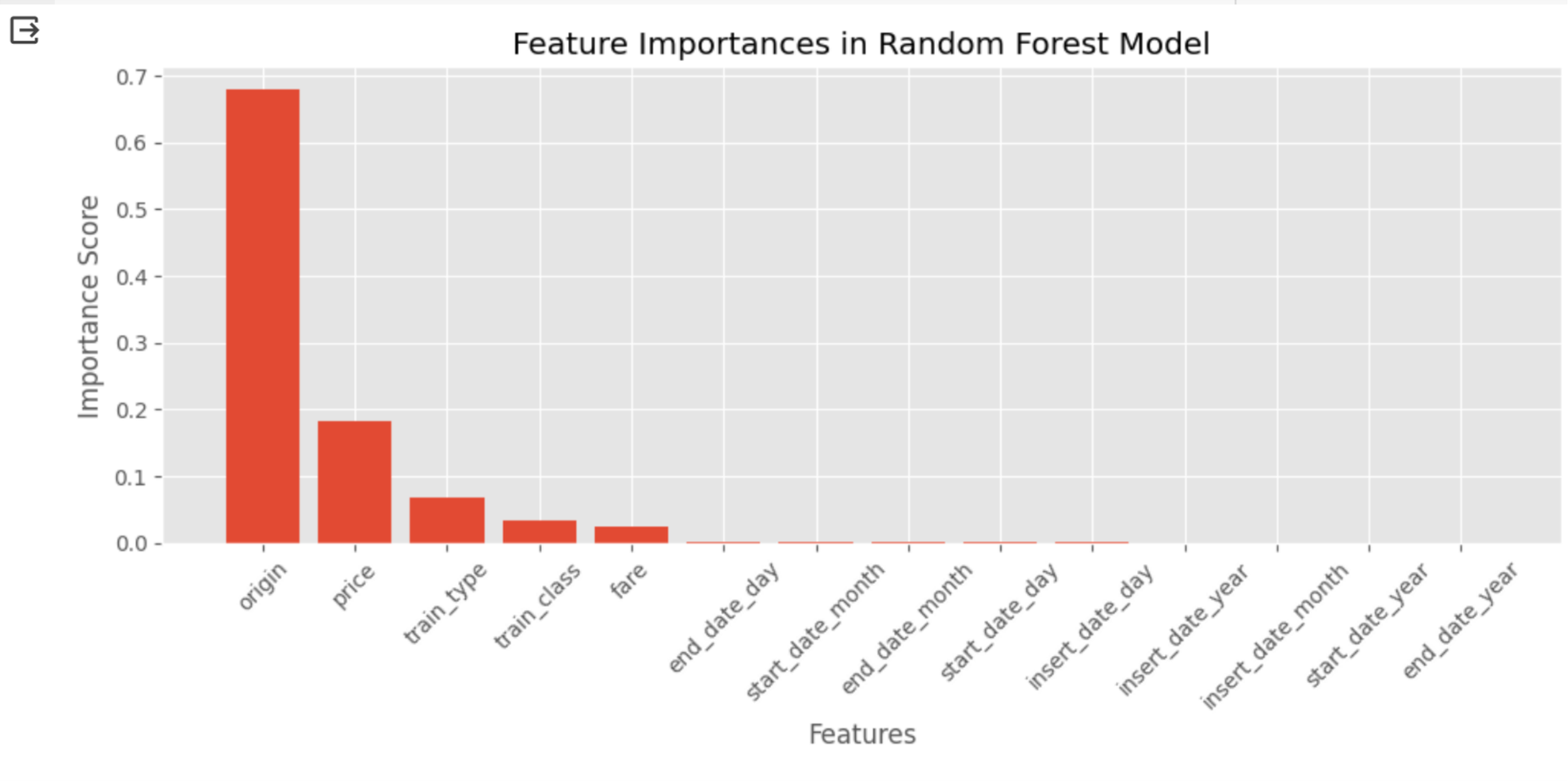
2.1. Random Forest Algorithm

Implement algorithm code

```python
import matplotlib.pyplot as plt

# Feature importances from the model
feature_importances = rf.feature_importances_

# Assuming X includes the preprocessed features of your dataset
feature_names = X.columns  # Ensure X has the correct columns after preprocessing
feature_importance_dict = dict(zip(feature_names, feature_importances))

# Sort features by importance
sorted_features = sorted(feature_importance_dict.items(), key=lambda x: x[1], reverse=True)

# Create a bar chart for feature importances
plt.figure(figsize=(10, 5))
plt.bar(range(len(sorted_features)), [val[1] for val in sorted_features], tick_label=[val[0] for val in sorted_features])
plt.xticks(rotation=45)
plt.xlabel('Features')
plt.ylabel('Importance Score')
plt.title('Feature Importances in Random Forest Model')
plt.tight_layout()
plt.show()

```

Plot Graph

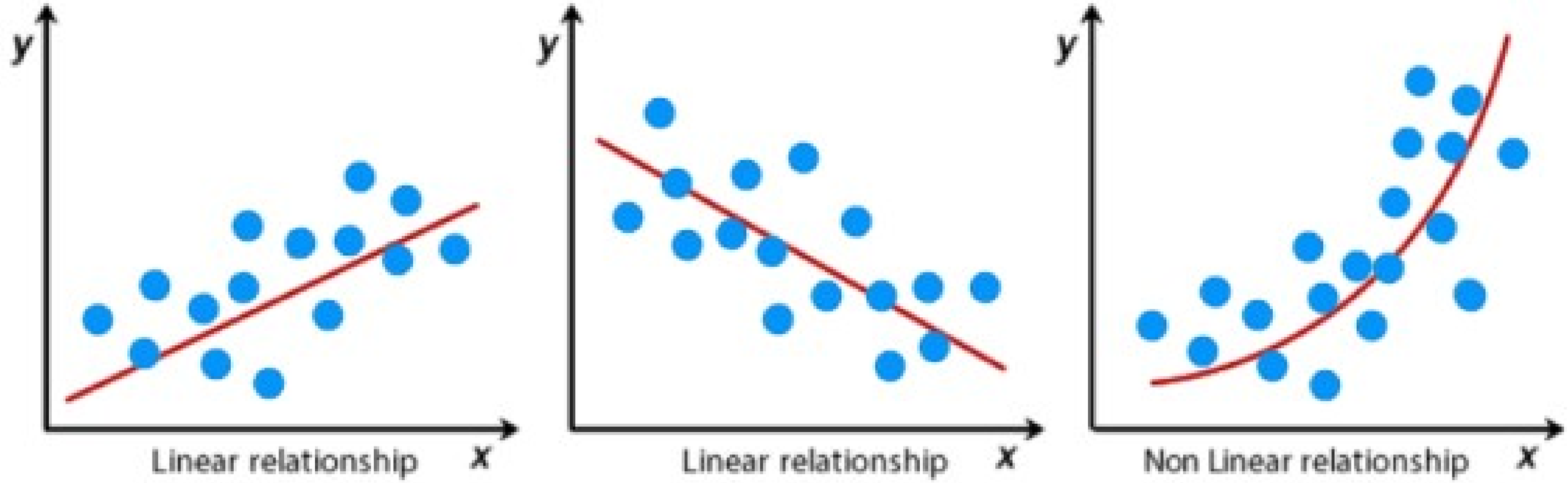
2.1. Random Forest Algorithm



Plot Graph

2.2. Linear Regression

Linear Regression aims to find the best-fitting line, minimizing the difference between observed and predicted values using least squares.



2.2. Linear Regression

The purpose of using linear regression in training models:
Linear regression is used to model and predict relationships between variables. Its purposes include making predictions, understanding variable impact
How to use linear Regression for training model:

- Importance of feature
- Prediction

2.2. Linear Regression

How i use my linear Regression:

- **Prediction**: it aims to predict the value such as amount of price, people, date with the dataset
- **Feature of Importance**: It helps identify which features contribute more significantly to the model's predictions, aiding in model interpretation, variable selection.

2.2. Linear Regression

Data Processing

Split data to 2 things which X is the predictor and y is the outcome

Library

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

```
# Extract features and target variable
X = df[features]
y = df[target]
```

2.2. Linear Regression

Implement Algorithm

Calling linear regression model and then fit my train data to the model.

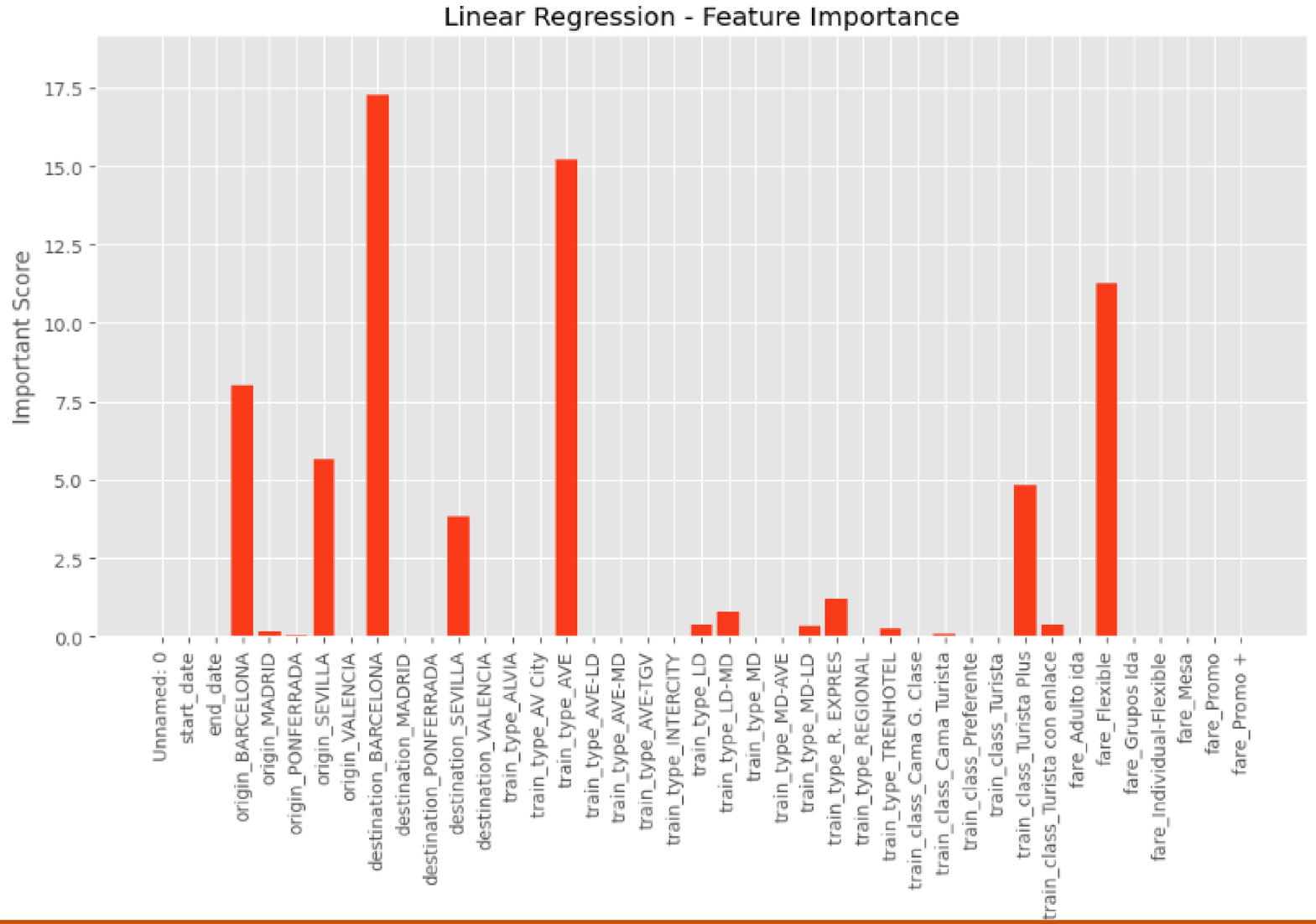
```
# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X, y)

# Predict the target variable based on the actual_duration
predictions = model.predict(X)
```

2.2. Linear Regression

Finding Coefficients to determine weight assigned to each feature in the model

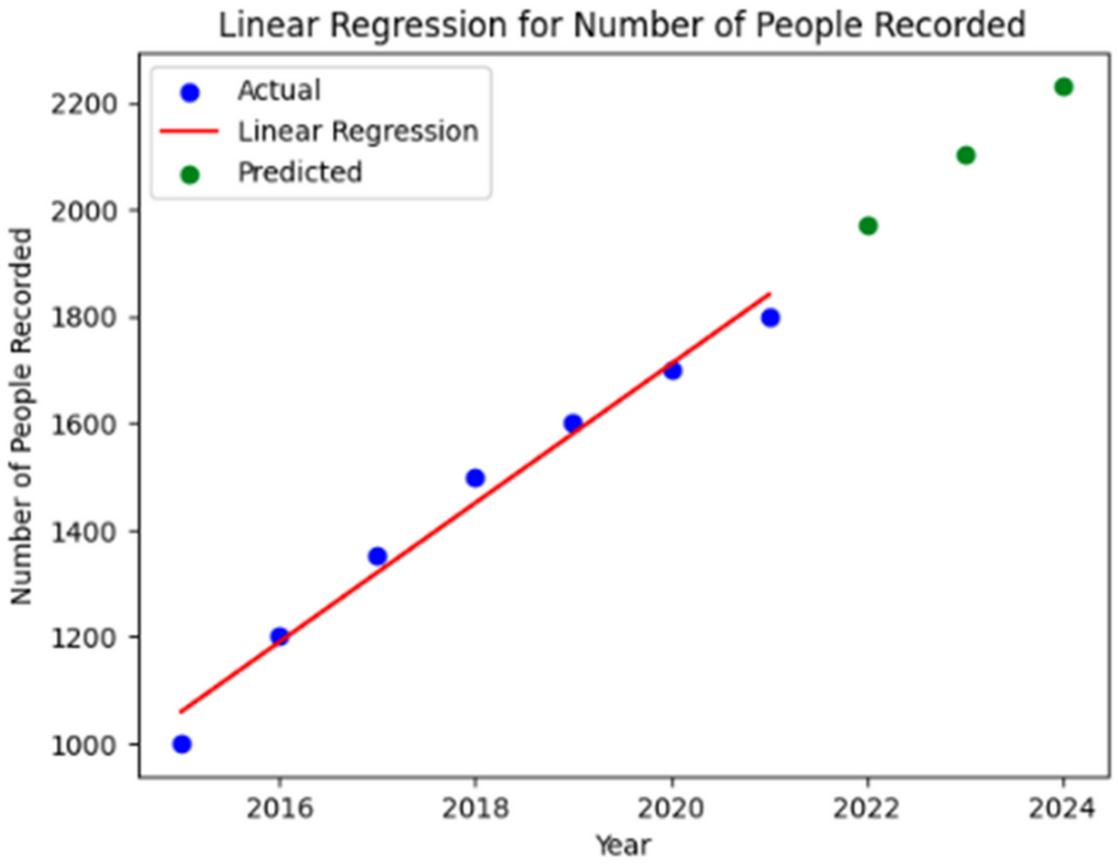
```
# Perform one-hot encoding on categorical columns
categorical_cols = ['origin', 'destination', 'train_type', 'train_class', 'fare']
X_encoded = pd.get_dummies(X, columns=categorical_cols)

# Fit the linear regression model
model = LinearRegression()
model.fit(X_encoded, y)

# Retrieve the coefficients
coefficients = model.coef_
```

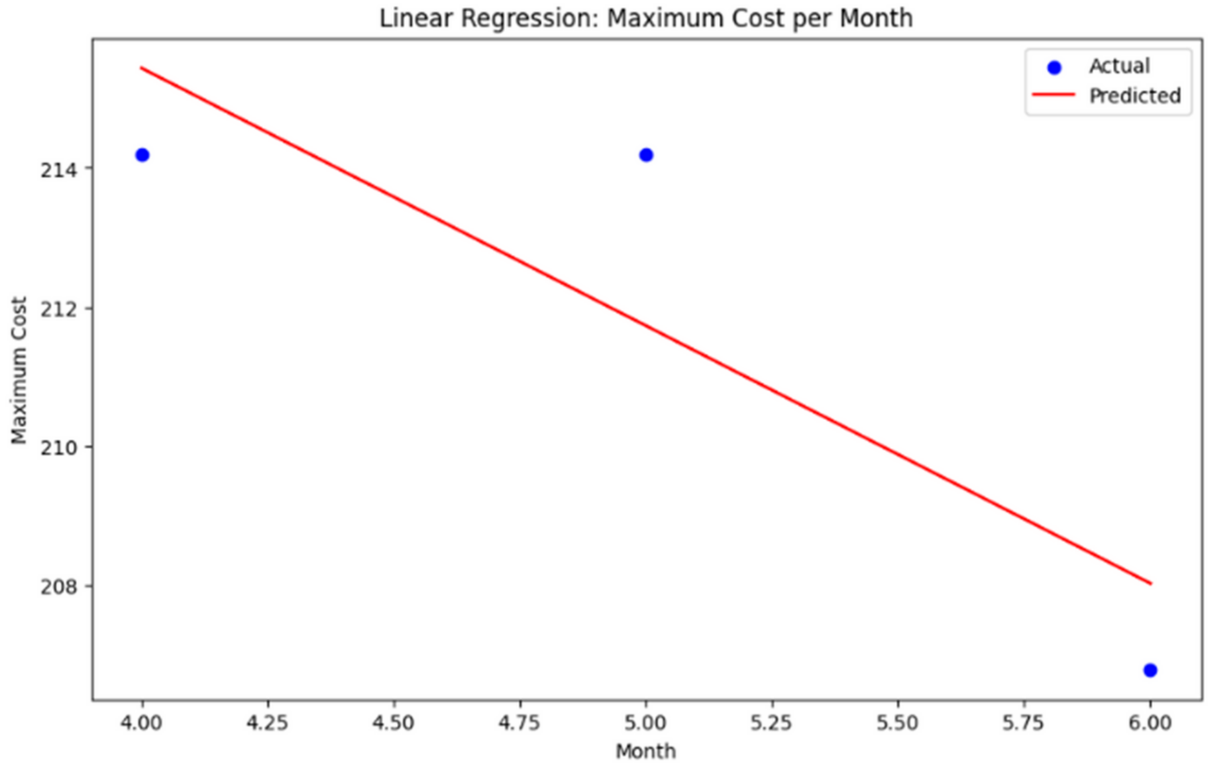
2.2. Linear Regression



2.2. Linear Regression

- **Result**





2.3. Decision Tree

The purpose is used for predictive modeling and classify data or what will come next

- Versatility
- Interpretability
- Non-linearity
- Feature Importance
- Handling Missing Value
- Ease of Use
- High Accuracy score

2.3. Decision Tree

➥ IMPLEMENTATION

```
# Selecting features and target
features = ['origin', 'train_type', 'price', 'train_class', 'fare',
            'insert_date_year', 'insert_date_month', 'insert_date_day',
            'start_date_year', 'start_date_month', 'start_date_day',
            'end_date_year', 'end_date_month', 'end_date_day']
X = data[features]
y = data['destination']
```

2.3. Decision Tree

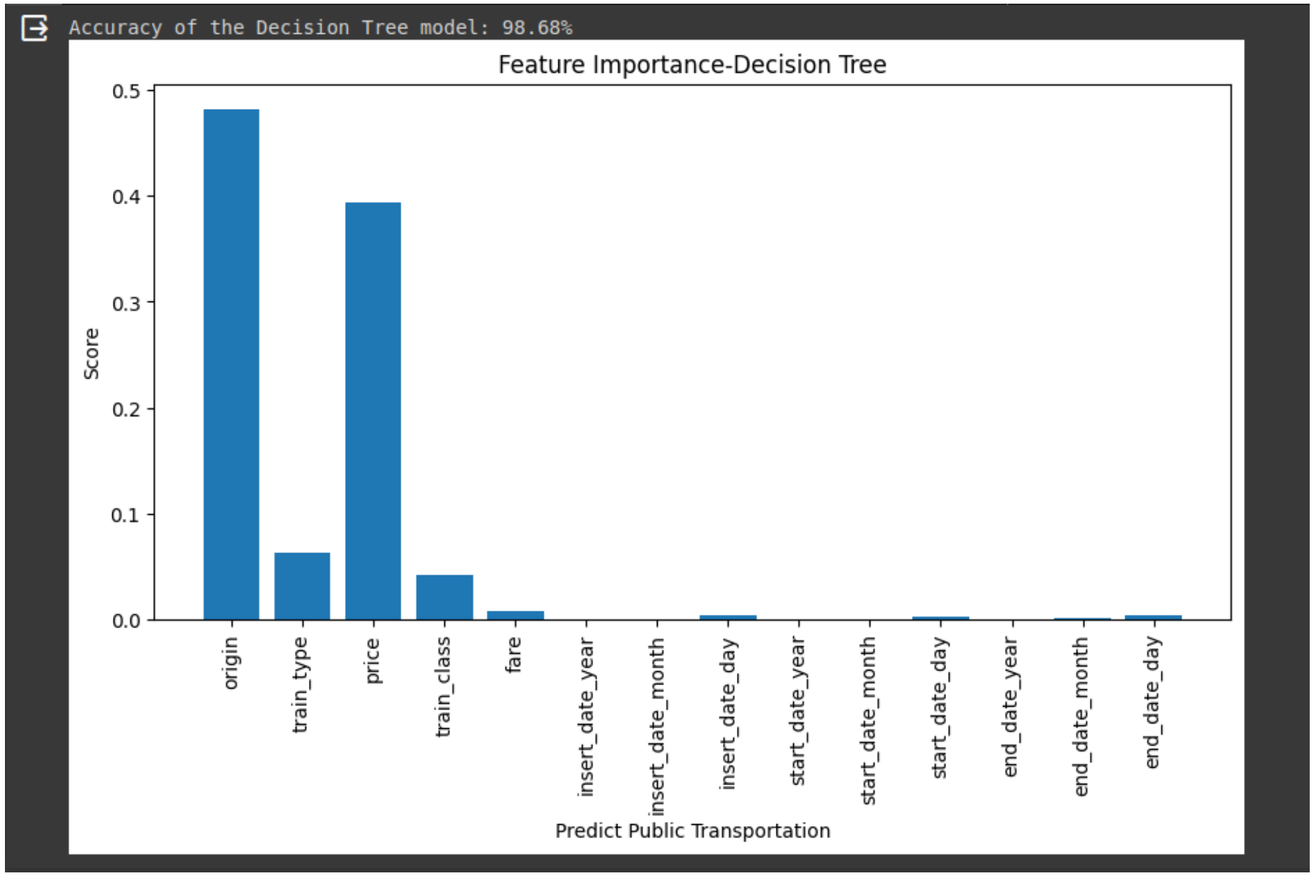
```
# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scaling the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the Decision Tree Classifier
decision_tree = DecisionTreeClassifier(random_state=42)

# Fit the model to the training data
decision_tree.fit(X_train, y_train)

# Predicting the Test set results
y_pred = decision_tree.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
accuracy_percentage = accuracy * 100

# Print the accuracy as a percentage
print(f'Accuracy of the Decision Tree model: {accuracy_percentage:.2f}%')
```

2.3. Decision Tree

```
# Plotting a graph (e.g., feature importance)
feature_importance = decision_tree.feature_importances_
plt.figure(figsize=(10, 5))

# plt.bar(range(len(feature_importance)), feature_importance)
plt.bar(range(len(features)), feature_importance)
plt.xticks(range(len(features)), features, rotation=90)
plt.xlabel('Predict Public Transportation')
plt.ylabel('Score')
plt.title('Feature Importance-Decision Tree')
plt.show()
```

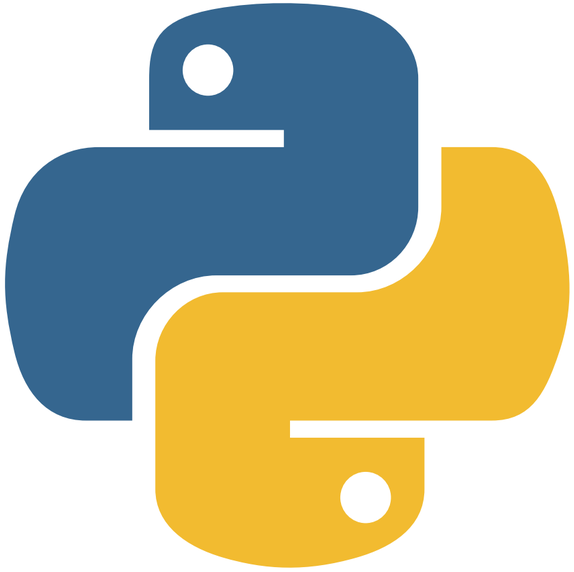
2.3. Decision Tree

➥ RESULT



2.4.Technologies and Tools

- Python Programming Language
- Kaggle for data collection
- Google Colab for IDE



kaggle

3. Evalutation of Each Model

To sum up, we can see that **Decision Tree** is more accurate than **Random Forest** since **Linear Regression** cannot make comparison because it is designed for predicting continuous numeric values, not classification tasks with accuracy percentages.

4. Conclusion

- Summary what we have done
- Compare each model
- Draw results of each model

6. References

- https://www.kaggle.com/datasets/northpatawee/spain-public-transportation
- https://scikit-learn.org/stable/

THANK YOU !