Prepared by Mohamed Fawzi

WEEK 4 OF CELLULA INTERNSHIP

Task'4

Fare Amount Prediction

Dataset Overview

- Contains trip details:
- Categorical: Car Condition, Weather, Traffic Condition
- Numerical: distance, bearing, jfk_dist, ewr_dist, lga_dist, passenger_count, time features
- Target: fare_amount
- After cleaning: 15 features.

Data Preprocessing Steps

- Dropped irrelevant columns (IDs, names, raw datetime).
- Converted categorical variables into numeric using One-Hot Encoding (0/1).

- Handled missing values and duplicates.
- Split data into Train/Test sets.

Data Preprocessing Steps

```python
cols_to_drop = [
    "key", "Driver Name", "pickup_datetime",
    "pickup_longitude", "pickup_latitude",
    "dropoff_longitude", "dropoff_latitude",
    "sol_dist", "nyc_dist" , "User ID" , "User Name"
]

df_clean = df.drop(columns=cols_to_drop)
```

```python
df_model = df_model.dropna()
```

```python
import pandas as pd



for col in df.columns:
    print("="*50)
    print(f"Column: {col}")
    print(f"Data Type: {df[col].dtype}")
    print(f"Number of Unique Values: {df[col].nunique()}")

    if df[col].dtype == 'object' or df[col].nunique() < 20:
        print(f"Classes/Unique values: {df[col].unique()}")
```

```python
categorical_cols = ["Car Condition", "Weather", "Traffic Condition"]

df_model = pd.get_dummies(df_clean, columns=categorical_cols, drop_first=True)

df_model = df_model.astype(int, errors='ignore')

df_model.head()
```

Modeling

- Baseline Model: Linear Regression
- R^2 Score: 0.26
- RMSE: 8.55
- Improved Model: Random Forest (with Hyperparameter Tuning)
- R^2 Score: 0.77
- RMSE: 0.20

```
print("R^2 Score:", r2_score(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
```

6] ✓ 0.0s

```
R^2 Score: 0.77
RMSE: 0.2
```

Hyperparameter Tuning

- Hyperparameters are model settings that are not learned automatically; we choose them before training.
- Examples for Random Forest:
- Number of trees (n_estimators)
- Maximum depth of trees (max_depth)
- Minimum samples to split a node (min_samples_split)
- Minimum samples per leaf (min_samples_leaf)

Hyperparameter Tuning

- Why Tuning?
- Default values may not give the best accuracy.
- Adjusting these values improves model performance.
-

Method Used:

- GridSearchCV
  - Tries different combinations of parameters.
  - Uses cross-validation to evaluate each combination.
  - Selects the best parameters based on R^2 score.

Result:

After tuning, Linear Regression performance improved significantly:

- R^2 Score: 0.77
- RMSE: 0.20

```
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
```

Hyperparameter Tuning

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

```
print("R^2 Score:", r2_score(y_test, y_pred))
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
```

```
R^2 Score: 0.77
RMSE: 0.2
```

Feature Importance

- Most important features:
- Distance
- Traffic Condition
- Weather
- Passenger count
- Bearing
-

Conclusion

- This project demonstrated how data-driven approaches can be used to predict taxi fare amounts accurately.
- After cleaning and preparing the dataset, we experimented with different models:
- Linear Regression provided a simple baseline but with limited accuracy.
- Random Forest with Hyperparameter Tuning significantly improved performance, achieving an R^2 score of 0.77 and an RMSE of 0.20.
- The model shows that distance, traffic conditions, weather, and passenger count are key factors affecting taxi fares.
- This work highlights the value of machine learning in building fair, efficient, and transparent fare estimation systems.

Thank you