April 28, 2025

[ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
 ↪precision_score, recall_score, f1_score, confusion_matrix
from google.colab import files

uploaded = files.upload()

df = pd.read_csv("trip_data.csv")

base_fare = 20
df["Fare"] = (
    base_fare
    + df["Trip_Distance"] * 8
    + df["Surge_Pricing_Flag"] * 10
    + df["Traffic_Level"] * 2
    + (5 - df["Driver_Rating"]) * 3
    + (5 - df["User_Rating"]) * 1.5
    + df["Weather_Condition"] * 2
)
df["Fare"] = df["Fare"].round(2)

categorical_features = ["Time_of_Day", "Weather_Condition"]
numerical_features = ["Trip_Distance", "Traffic_Level", "Driver_Rating",␣
 ↪"User_Rating"]

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])
```

```
numerical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="mean")),
    ("scaler", StandardScaler())
])

# --------- Ride Fare Prediction (Linear Regression) ---------
X_fare = df.drop(columns=["Fare"])
y_fare = df["Fare"]

preprocessor_fare = ColumnTransformer(
    transformers=[
        ("cat", categorical_transformer, categorical_features)
    ],
    remainder="passthrough"
)

model_fare = Pipeline(steps=[
    ("preprocessor", preprocessor_fare),
    ("regressor", LinearRegression())
])

X_train_fare, X_test_fare, y_train_fare, y_test_fare = train_test_split(X_fare,␣
 ↪y_fare, test_size=0.2, random_state=42)

model_fare.fit(X_train_fare, y_train_fare)

y_pred_fare = model_fare.predict(X_test_fare)

rmse = np.sqrt(mean_squared_error(y_test_fare, y_pred_fare))
r2 = r2_score(y_test_fare, y_pred_fare)

print("----- Ride Fare Prediction -----")
print("RMSE:", rmse)
print("R2 Score:", r2)

# --------- Surge Pricing Prediction (Logistic Regression) ---------
X_surge = df.drop(columns=["Surge_Pricing_Flag", "Fare"])
y_surge = df["Surge_Pricing_Flag"]

preprocessor_surge = ColumnTransformer(
    transformers=[
        ("cat", categorical_transformer, categorical_features),
        ("num", numerical_transformer, numerical_features)
    ]
)

model_surge = Pipeline(steps=[
```

```
    ("preprocessor", preprocessor_surge),
    ("classifier", LogisticRegression())
])

X_train_surge, X_test_surge, y_train_surge, y_test_surge =␣
 ↪train_test_split(X_surge, y_surge, test_size=0.2, random_state=42)

param_grid = {'classifier__C': [0.01, 0.1, 1, 10, 100]}
grid_search = GridSearchCV(model_surge, param_grid, cv=2, scoring='f1')
grid_search.fit(X_train_surge, y_train_surge)

best_model_surge = grid_search.best_estimator_

y_pred_surge = best_model_surge.predict(X_test_surge)

accuracy = accuracy_score(y_test_surge, y_pred_surge)
precision = precision_score(y_test_surge, y_pred_surge)
recall = recall_score(y_test_surge, y_pred_surge)
f1 = f1_score(y_test_surge, y_pred_surge)
conf_matrix = confusion_matrix(y_test_surge, y_pred_surge)

print("\n----- Surge Pricing Prediction -----")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:\n", conf_matrix)
```

```
<IPython.core.display.HTML object>
```

```
Saving trip_data.csv to trip_data (1).csv
----- Ride Fare Prediction -----
RMSE: 1.0048591735576161e-13
R2 Score: 1.0

----- Surge Pricing Prediction -----
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Confusion Matrix:
 [[1 0]
 [0 1]]
```