Uber Fare Prediction - Complete Viva Guide

Project Overview

This project predicts Uber ride fares using machine learning based on ride distance calculated from GPS coordinates.

Code Explanation with Viva Questions

1 Importing Libraries

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

What it does: Imports necessary libraries for data handling, machine learning, and evaluation.

Viva Questions:

Q1: Why do we use pandas?
A: Pandas is used for loading, manipulating, and analyzing structured data in DataFrame format (like tables).

Q2: What is the purpose of scikit-learn?
A: Scikit-learn provides tools for machine learning including models, data splitting, and evaluation metrics.

Q3: Why numpy?
A: NumPy is used for mathematical operations and array calculations, especially in the Haversine formula.

---

2 Loading Dataset

```python
df = pd.read_csv("uber.csv")
```

What it does: Loads the Uber dataset from CSV file into a pandas DataFrame.

Viva Questions:

Q4: What is a DataFrame?
A: A DataFrame is a 2-dimensional labeled data structure with rows and columns, similar to an Excel spreadsheet.

Q5: What columns does the Uber dataset typically contain?
A: Pickup/dropoff latitude and longitude, fare amount, passenger count, pickup datetime, etc.

Q6: Why CSV format?
A: CSV (Comma Separated Values) is a simple, universal format for storing tabular data that can be easily read by many programs.

---

3 Feature Engineering - Haversine Distance

```
def haversine(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = np.sin(dlat/2)**2 + np.cos(lat1)*np.cos(lat2)*np.sin(dlon/2)**2
    return 6371 * (2*np.arcsin(np.sqrt(a)))

df["distance_km"] = haversine(df["pickup_longitude"], df["pickup_latitude"],
                              df["dropoff_longitude"], df["dropoff_latitude"])
```

What it does: Calculates the straight-line distance between pickup and dropoff locations in kilometers.

Viva Questions:

Q7: What is the Haversine formula?
A: It's a formula to calculate the great-circle distance between two points on a sphere (Earth) using their latitude and longitude.

Q8: Why can't we use simple Euclidean distance?
A: Because Earth is spherical, not flat. Euclidean distance would give incorrect results, especially for longer distances.

Q9: What does 6371 represent?
A: It's Earth's radius in kilometers. Multiplying by this converts the angular distance to actual kilometers.

Q10: Why convert to radians?
A: Trigonometric functions (sin, cos) in programming work with radians, not degrees.

Q11: What is feature engineering?
A: Creating new features from existing data to improve model performance. Here, we created distance_km from GPS coordinates.

---

4 Data Cleaning - Removing Outliers

```
df = df[(df["fare_amount"] > 0) & (df["distance_km"] > 0) & (df["distance_km"] < 100)]
```

What it does: Filters out invalid and extreme values from the dataset.

Viva Questions:

Q12: Why remove zero or negative fares?
A: These are invalid/erroneous entries. Uber cannot charge zero or negative money for a ride.

Q13: Why remove distances $>$ 100 km?
A: These are likely data errors or outliers. Most Uber rides are shorter; very long distances would skew the model.

Q14: What are outliers?
A: Extreme values that differ significantly from other observations. They can negatively impact model training.

Q15: What happens if we don't remove outliers?
A: The model might learn incorrect patterns, leading to poor predictions on normal data.

---

5 Defining Features and Target

```
X = df[["distance_km"]]
y = df["fare_amount"]
```

What it does: Separates input feature (distance) from output target (fare).

Viva Questions:

Q16: What is X and y in machine learning?
A: X represents input features (independent variables), y represents the target/output (dependent variable) we want to predict.

Q17: Why double brackets [[...]] for X?
A: To maintain DataFrame structure. Single bracket would return a Series; double bracket returns a DataFrame.

Q18: Can we use multiple features?
A: Yes! We could add passenger_count, time_of_day, day_of_week, etc., as additional features.

---

6 Splitting Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

What it does: Splits data into 80% training and 20% testing sets.

Viva Questions:

Q19: Why split data into train and test?
A: To train the model on one portion and evaluate its performance on unseen data, preventing overfitting.

Q20: What is test_size=0.2?
A: 20% of data is kept for testing, 80% for training. This is a common split ratio.

Q21: What does random_state=42 do?
A: It ensures reproducibility. The same random split occurs every time you run the code.

Q22: What is overfitting?
A: When a model learns training data too well (including noise) and performs poorly on new data.

---

7 Model 1: Linear Regression

```
lr = LinearRegression()
lr.fit(X_train, y_train)
pred_lr = lr.predict(X_test)
```

What it does: Trains a linear regression model and makes predictions.

Viva Questions:

Q23: What is Linear Regression?
A: A supervised learning algorithm that models the relationship between features and target as a straight line: y = mx + b.

Q24: How does Linear Regression work?
A: It finds the best-fit line by minimizing the sum of squared errors between predicted and actual values.

Q25: When is Linear Regression suitable?
A: When there's a linear relationship between input and output variables.

Q26: What does .fit() do?
A: It trains the model by learning the relationship between X_train and y_train.

Q27: What does .predict() do?
A: It uses the trained model to predict y values for new X values (X_test).

---

8 Model 2: Random Forest

```
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
pred_rf = rf.predict(X_test)
```

What it does: Trains a Random Forest model and makes predictions.

Viva Questions:

Q28: What is Random Forest?
A: An ensemble learning method that builds multiple decision trees and combines their predictions.

Q29: How is Random Forest different from Linear Regression?
A: Linear Regression assumes linear relationships; Random Forest can capture non-linear patterns and complex interactions.

Q30: What is a decision tree?
A: A tree-like model that makes decisions by splitting data based on feature values.

Q31: Why is it called "Random Forest"?
A: It creates a "forest" of many decision trees, each trained on random subsets of data and features.

Q32: What is ensemble learning?
A: Combining multiple models to produce better predictions than any single model.

---

9 Model Evaluation

```
print("Linear Regression R2:", r2_score(y_test, pred_lr))
print("Linear Regression RMSE:", mean_squared_error(y_test, pred_lr, squared=False))

print("Random Forest R2:", r2_score(y_test, pred_rf))
print("Random Forest RMSE:", mean_squared_error(y_test, pred_rf, squared=False))
```

What it does: Evaluates both models using R² and RMSE metrics.

Viva Questions:

Q33: What is R² (R-squared)?
A: Coefficient of determination that measures how well the model fits the data. Range: 0 to 1 (1 = perfect fit).

Q34: How to interpret R² score?
A: - R² = 0.8 means 80% of variance in fare is explained by distance - Higher R² = better model fit

Q35: What is RMSE?
A: Root Mean Squared Error - the average magnitude of prediction errors in the same units as the target (dollars).

Q36: What is a good RMSE value?
A: Lower is better. If RMSE = 3, predictions are off by ~$3 on average.

Q37: Why use both R^2 and RMSE?
A: R^2 shows relative fit quality; RMSE shows absolute error magnitude. Together they give a complete picture.

Q38: What is MSE vs RMSE?
A: MSE (Mean Squared Error) is in squared units. RMSE is the square root of MSE, giving interpretable units.

Project Summary Questions

Q39: Summarize your project in 2 minutes.
A: "We built a machine learning system to predict Uber fares. First, we calculated ride distance using GPS coordinates with the Haversine formula. After cleaning the data by removing invalid entries and outliers, we trained two models: Linear Regression (simple, linear relationship) and Random Forest (complex, non-linear patterns). We evaluated both using R^2 and RMSE metrics. The model with higher R^2 and lower RMSE performs better."

Q40: Which model performed better?
A: (Check your output!) Typically Random Forest performs better because it can capture non-linear relationships, but Linear Regression is simpler and faster.

Q41: What are the limitations of your model?
A: - Only uses distance; ignores time of day, traffic, weather, surge pricing - Assumes straight-line distance, not actual road distance - May not generalize to other cities

Q42: How could you improve this model?
A: - Add more features: time, day, weather, traffic, pickup/dropoff zones - Use actual road distance (via APIs) - Try other models: Gradient Boosting, XGBoost, Neural Networks - Perform hyperparameter tuning

Q43: What is supervised learning?
A: Learning from labeled data where we know the correct output (fare amount) for each input (distance).

Q44: What is regression vs classification?
A: - Regression: Predicting continuous values (fare amount) - Classification: Predicting categories (will it rain: yes/no)

Q45: What is the difference between training and testing?
A: - Training: Model learns patterns from training data - Testing: We evaluate how well the model predicts on unseen test data

---

Advanced Questions

Q46: What is cross-validation?
A: A technique that splits data into multiple folds, trains on some and tests on others, rotating through all combinations for robust evaluation.

Q47: What is bias-variance tradeoff?
A: - Bias: Error from oversimplifying the model (underfitting) - Variance: Error from model being too complex (overfitting) - Goal: Balance both for optimal performance

Q48: What hyperparameters could you tune in Random Forest?
A: n_estimators (number of trees), max_depth (tree depth), min_samples_split, min_samples_leaf.

Q49: What is feature scaling? Do you need it here?
A: Normalizing features to the same scale. Not needed for tree-based models (Random Forest) but helpful for Linear Regression if using multiple features with different scales.

Q50: What is the curse of dimensionality?
A: As you add more features, you need exponentially more data to maintain model performance.

Quick Reference Table

Metric	Meaning	Good Value
R²	Model fit quality	Closer to 1.0
RMSE	Average prediction error	Lower is better
Train/Test Split	80/20 or 70/30	Industry standard

Final Tips for Viva

1. **Understand the flow:** Data → Feature Engineering → Cleaning → Split → Train → Evaluate
2. **Know the "why":** Why each step matters
3. **Be honest:** If you don't know something, say so and explain your thinking
4. **Connect concepts:** Link machine learning terms to real-world examples

5. **Practice explaining:** Use simple language, avoid just memorizing definitions

---

Good luck with your viva!