DA2011 Interim Discussion Worksheet (Updated)

Ashoka Wijesinghe (24ada006)

- Task: Task 1 – Problem Definition & Scope
- Speaking Script:

Our central prediction problem focuses on estimating the Historical Cost of Ride based on features such as ride duration, vehicle type, and location. This dataset allows us to model pricing behavior using only internal ride data.

- Supporting Code:

```
# Import dataset
import pandas as pd
df = pd.read_csv("dynamic_pricing.csv")

# Define features and target
X = df.drop(columns=['Historical_Cost_of_Ride'])
y = df['Historical_Cost_of_Ride']
```

- Key Takeaway: Defines the prediction focus clearly — no external data used.

Anusha Dassanayake (24ada008)

- Task: Task 2 – Objectives & Evaluation
- Speaking Script:

The primary objective is to predict ride cost accurately. Secondary objectives include selecting the most predictive features and comparing regression models. We will evaluate models using R^2, RMSE, and MAE metrics.

- Supporting Code:

```
# Define evaluation metrics
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

# Example metrics calculation
r2 = r2_score(y_true, y_pred)
rmse = mean_squared_error(y_true, y_pred, squared=False)
mae = mean_absolute_error(y_true, y_pred)
```

- Key Takeaway: Model success will be measured using R^2, RMSE, and MAE.

R.W.A.B.D. Jayasekara (24ada048)

- Task: Task 3 – Data Cleaning & Integrity
- Speaking Script:

We verified that the dataset contained no missing or duplicate values. The data was then split into training and testing sets using an 80/20 ratio. All exploratory data analysis (EDA) was performed only on the training dataset to avoid leakage.

- Supporting Code:

```
# Check for missing or duplicate values
df.isnull().sum()
df.duplicated().sum()

# Split data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

- Key Takeaway: Data integrity confirmed; EDA done only on training set.

H.M.S.K. Jayathilaka (24ada033)

- Task: Task 3 – Numerical EDA Findings
- Speaking Script:

The distribution of the target variable, Historical_Cost_of_Ride, was approximately normal. Correlation analysis revealed that Expected_Ride_Duration had the strongest positive correlation with ride cost, around 0.93, confirming its predictive importance.

- Supporting Code:

```
# Correlation heatmap
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

- Key Takeaway: Expected_Ride_Duration shows the strongest correlation (~0.93) with cost.

B.N. Lakshan (24ada037)

- Task: Task 3 – Visual EDA & Interpretations
- Speaking Script:

In our visual exploratory data analysis, we identified strong relationships between expected ride duration and historical cost, and between vehicle type and cost. Premium vehicles consistently showed higher ride costs compared to economy rides, confirming pricing differentiation.

- Supporting Code:

```
# Visual EDA – Ride Duration vs Cost and Vehicle Type vs Cost
import seaborn as sns
import matplotlib.pyplot as plt

# Ride Duration vs Cost
sns.scatterplot(data=df, x='Expected_Ride_Duration',
y='Historical_Cost_of_Ride')
plt.title('Ride Duration vs Cost')
plt.show()

# Vehicle Type vs Cost
sns.boxplot(data=df, x='Vehicle_Type', y='Historical_Cost_of_Ride')
plt.title('Vehicle Type vs Cost')
plt.show()
```

- Key Takeaway: Visual EDA confirmed strong links between ride duration and cost, and between vehicle type and pricing levels.

K.R.D. Alwis (24ada034)

- Task: Task 2/3 – Modeling Plan & Next Steps
- Speaking Script:

Our modeling plan involves comparing Linear Regression, Decision Tree Regression, and Random Forest Regression. We will use Best Subset Selection to identify the most relevant predictors and perform cross-validation for robustness. The data is now ready for model training in Task 4.

- Supporting Code:

```
# Example model setup
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(max_depth=5),
    'Random Forest': RandomForestRegressor(n_estimators=100,
random_state=42)
}
```

- Key Takeaway: Ready for model building using selected predictors and cross-validation.