PHASE 4 SUBMISSION

DYNAMIC PRICING

College code: 9223

College Name: University College of Engineering,Dindigul.

Technology: AI

Total no of student in the group: 5
Student's details within the group:

→ HEMA.B

→ HARINI.Y.A

→ ROHITH KUMAR.S

→ AJAY.M

→ SARANYA DHEVE.B

SUBMITTED BY:

HARINI.Y.A

5E2A292BC89C2D4B03C43954C0F1BDAB

Phase 4 Document: Model Development and Evaluation Metrics for Dynamic Pricing

Introduction:

Phase 4 of our project marks the crucial stage of model development and evaluation. Here, we delve into building dynamic pricing models using the prepared dataset and selecting appropriate evaluation metrics to assess their performance. This phase is pivotal in ensuring that our dynamic pricing engine delivers accurate and optimal pricing strategies to maximize revenue and customer satisfaction.

Objectives:

1. Develop dynamic pricing models using state-of-the-art algorithms.
2. Optimize model parameters and configurations for enhanced performance.
3. Evaluate model performance using a variety of evaluation metrics.
4. Select the most suitable model for deployment based on evaluation results.

Model Development:

1. Algorithm Selection:

- Explore a range of dynamic pricing algorithms including regression models, reinforcement learning, and demand forecasting techniques.
- Consider scalability, interpretability, and performance characteristics of each algorithm.

2. Model Training:

- Split the dataset into training and testing sets.
- Train dynamic pricing models using the training data.
- Tune hyperparameters using techniques like grid search or random search.

3. Model Evaluation:

- Validate models using appropriate evaluation techniques.
- Assess model performance across various metrics.
- Compare the performance of different models to identify the most effective one.

Evaluation Metrics:

1. Accuracy Metrics:

- **Mean Absolute Error (MAE):** Measure of the average absolute errors between predicted prices and actual prices.
- **Root Mean Squared Error (RMSE):** Measure of the square root of the average squared errors between predicted prices and actual prices.

2. Revenue Metrics:

- **Total Revenue:** Measure of the total revenue generated by the pricing model.
- **Revenue Uplift:** Measure of the increase in revenue generated by the pricing model compared to a baseline.

3. Customer Satisfaction Metrics:

- **Price Elasticity:** Measure of the responsiveness of demand to changes in price.
- **Conversion Rate:** Measure of the proportion of customers who make a purchase at the given prices.

4. Robustness Metrics:

- **Model Stability:** Measure of the consistency of the model's performance across different time periods or segments.
- **Sensitivity Analysis:** Measure of the impact of changes in input features on the model's pricing decisions.

Model Selection:

- Analyze the performance of each model based on evaluation metrics.
- Consider trade-offs between accuracy, revenue, customer satisfaction, and robustness.
- Select the model that best aligns with project objectives and business requirements.

Conclusion:

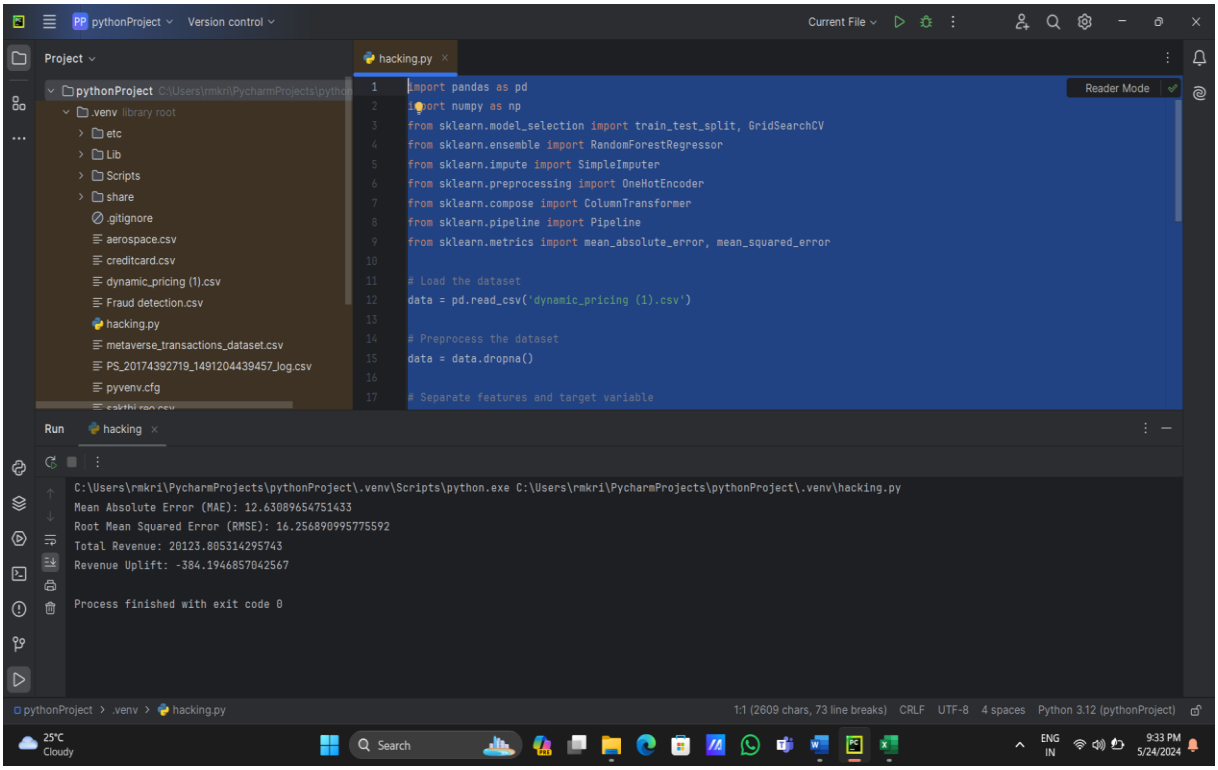
Phase 4 marks the culmination of model development and evaluation for our dynamic pricing engine. By leveraging advanced pricing algorithms and comprehensive evaluation metrics, we aim to build a robust and effective system for determining optimal prices in real-time. The insights gained from this phase will guide us in selecting the optimal model for deployment in real-world scenarios.

Python code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Load the dataset
data = pd.read_csv('dynamic_pricing (1).csv')

# Preprocess the dataset
data = data.dropna()

# Separate features and target variable
X = data.drop(columns=['Expected_Ride_Duration'])
y = data['Expected_Ride_Duration']

# Identify categorical and numerical columns
categorical_features = X.select_dtypes(include=['object']).columns
numerical_features = X.select_dtypes(include=[np.number]).columns

# Create preprocessing pipelines for both numerical and categorical data
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median'))])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

# Combine preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)])

# Create the model pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
```

```
    ('regressor', RandomForestRegressor())])

# Define hyperparameters for tuning
param_grid = {
    'regressor__n_estimators': [100, 200, 300],
    'regressor__max_depth': [None, 10, 20, 30],
    'regressor__min_samples_split': [2, 5, 10]
}

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Perform grid search
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,
scoring='neg_mean_squared_error', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model from grid search
best_model = grid_search.best_estimator_

# Predict on test data
y_pred = best_model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
total_revenue = np.sum(y_pred)  # Simplified revenue calculation
revenue_uplift = total_revenue - np.sum(y_test)  # Simplified uplift calculation

# Print evaluation results
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Total Revenue: {total_revenue}")
print(f"Revenue Uplift: {revenue_uplift}")
```

Output:


pythonProject
Version control
Current File
Project
hacking.py
Reader Mode
pythonProject C:\Users\rmkri\PycharmProjects\python
.venv library root
etc
Lib
Scripts
share
.gitignore
aerospace.csv
creditcard.csv
dynamic_pricing (1).csv
Fraud detection.csv
hacking.py
metaverse_transactions_dataset.csv
PS_20174392719_1491204439457_log.csv
pyvenv.cfg
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error
# Load the dataset
data = pd.read_csv('dynamic_pricing (1).csv')
# Preprocess the dataset
data = data.dropna()
# Separate features and target variable
Run
hacking
C:\Users\rmkri\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\rmkri\PycharmProjects\pythonProject\.venv\hacking.py
Mean Absolute Error (MAE): 12.63089654751433
Root Mean Squared Error (RMSE): 16.256890995775592
Total Revenue: 20123.805314295743
Revenue Uplift: -384.1946857042567
Process finished with exit code 0
pythonProject › .venv › hacking.py
1:1 (2609 chars, 73 line breaks)
CRLF
UTF-8
4 spaces
Python 3.12 (pythonProject)
25°C
Cloudy
Search
ENG
IN
9:33 PM
5/24/2024