Sprint 5

Day 10 Task 1:

Sprints for AI
Use Case For Demand Forecasting Using POS Transaction Data

Team Members:

Tufail Irshad
Amjad Ali Malik
Zameer Ahmad Mir
Asif Ahmad Najar
Irfan Ahmad Mutoo

Sprint 5 Day 10 Task 1

Ai Deployment and Validation Report

Introduction:

Overview

generated an POS transaction data of 10 thousand rows and then applied a machine learning model on that data set to predict demand of an item in feature. And then after, Deploying a machine learning model to analysis the demand of an item at a particular time period.

Data Preparation

- **Data collection: -**

Generated POS transaction data by using faker lib, including transaction amount, timestamp, items, product details, and mode of transaction etc.

```
!pip install faker
from faker import Faker
import random
from datetime import datetime, timedelta
import csv
print("Done")

# Create a Faker instance
fake = Faker()

# Generate synthetic data
def generate_synthetic_data(num_entries):
    synthetic_data = []
    for _ in range(num_entries):
        # Generate random date within the last year
        transaction_date = fake.date_between(start_date='-1y',
end_date='today')
        # Generate item purchased
        item_purchased = fake.word()
        # Generate random unit price
        unit_price = round(random.uniform(1, 1000), 2)
        # Generate random tax amount
        tax = round(random.choice([0.1, 0.3]),2)
```

```
        # Calculate transaction amount
        transaction_amount = round(unit_price + (unit_price * (tax / 100)), 2)
        # Generate random transaction mode
        transaction_mode = random.choice(['Cash', 'Credit Card', 'Debit Card'])
        # Generate random transaction currency
        transaction_currency = random.choice(['USD', 'INR'])
        # Append data to the synthetic data list
        synthetic_data.append([
            transaction_date.strftime('%Y-%m-%d'),
            item_purchased,
            unit_price,
            tax,
            transaction_amount,
            transaction_mode,
            transaction_currency
        ])
    return synthetic_data

# Number of synthetic data points to generate
num_entries = 10000

# Generate synthetic data
synthetic_data = generate_synthetic_data(num_entries)

# Write synthetic data to a CSV file
output_file = 'synthetic_data.csv'
with open(output_file, 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Date of Transaction', 'Item Purchased', 'Unit Price', 'Tax', 'Transaction Amount', 'Transaction Mode', 'Transaction Currency'])
    writer.writerows(synthetic_data)

print(f"Synthetic data written to {output_file}")
```

Data Preprocessing

- **Loading the Data**

  The initial step involves loading your cleaned DataFrame (df_cleaned). This DataFrame is assumed to have no missing values or outliers.

```
import pandas as pd
```

```
dff=pd.read_csv('/content/pos_dataset.csv')
dff.head(5)
```

- **Encoding Categorical Columns**

Categorical columns are encoded using OneHotEncoder.

```
# Columns to encode and scale
categorical_columns = ['Item Purchased', 'Transaction Mode',
'Transaction Currency']
# Encode categorical columns
def encode_categorical_columns(df, categorical_cols):
    # Initialize the OneHotEncoder
    ohe = OneHotEncoder(sparse=False, drop='first')
    encoded_df =
pd.DataFrame(ohe.fit_transform(df[categorical_cols]),
columns=ohe.get_feature_names_out(categorical_cols))

    # Concatenate the encoded columns back with the original
DataFrame
    df = df.drop(categorical_cols, axis=1).reset_index(drop=True)
    df = pd.concat([df, encoded_df], axis=1)
    return df
```

- **Data cleaning**

Address missing values, outliers, and inconsistencies in the dataset through techniques such as outlier removal, and data transformation.

```
# Check for missing value
print(dff.isnull().sum())
Date of Transaction       0
Item Purchased            0
Quantity                 30
Unit Price                2
Tax                       0
Transaction Amount        0
Transaction Mode         25
Transaction Currency      2
Total Amount In INR       0
```

- **Filling Missing Values**

```
# Find the most frequent value (mode) of the "Quantity" column
mode_quantity = dff["Quantity"].value_counts().idxmax()
```

```
# Fill missing values in the "Quantity" column with the mode
dff["Quantity"].fillna(mode_quantity, inplace=True)
```

Feature Engineering

Extract relevant features from the raw data, such as transaction amount, item purchased, date of transaction , mode of transaction.

Model Training and Evaluation

- **Splitting the Data**

Split the data into training and test sets

```
from sklearn.model_selection import train_test_split

df_preprocessed = pd.read_csv('preprocessed_data.csv')

# Split the data into features (X) and target (y)
X = df_preprocessed.drop(['Transaction Amount','Date of
Transaction'], axis=1)
y = df_preprocessed['Transaction Amount']
```

- **Training Multiple Models**

Train and evaluate multiple models to identify the best performer.

```
import pandas as pd
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "Support Vector Machine": SVR(),
}
results = {}
```

Error Calculation

Calculate the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to assess model accuracy.

```
for model_name, metrics in results.items():
    print(f"Model: {model_name}")
    print(f"  MSE: {metrics['MSE']}")
    print(f"  RMSE: {metrics['RMSE']}")
    print(f"  R-squared: {metrics['R-squared']}")
    print()

# Select the best model based on RMSE (or any other preferred
metric)
best_model_name = min(results, key=lambda x: results[x]['RMSE'])
best_model = models[best_model_name]

print(f"Best Model: {best_model_name}")
```

```
Model: Linear Regression
MSE: 5.494355140883775e-09
RMSE: 7.41239174685457e-05
```

Deployment : -

- **Model Integration**

 Integrate the trained model into the POS system or analytics platform for real-time or batch processing of transaction data.

- **Scalability**

-Ensure that the deployed model can handle large volumes of transactions efficiently and scale as the business grows.

- **API Development**

Develop APIs or microservices to facilitate seamless interaction between the deployed model and other systems or applications.

Validation

- **Data Quality Assurance**

  Continuously monitor the quality and integrity of the input data to ensure that the model receives accurate and reliable information . Model Performance

- **Monitoring**

  Monitor the deployed model's performance over time, tracking key performance indicators (KPIs) and comparing them against baseline metrics.

- **Feedback Loop**

  Establish a feedback loop to collect user feedback, address model drift, and incorporate new insights or features into future iterations of the model.

Security and Compliance

- **Data Privacy**

  Implement measures to protect sensitive customer information and comply with data privacy regulations (e.g., GDPR, CCPA).

- **Fraud Detection**

  Implement fraud detection algorithms to identify and mitigate potential security threats, such as fraudulent transactions or data breaches.

- **Regulatory Compliance**

  Ensure that the deployed model adheres to relevant industry regulations and standards, such as PCI-DSS for payment card security.

Conclusion

Summarize the key findings and outcomes of the AI deployment and validation process, highlighting improvements in decision-making, operational efficiency, and risk management. Provide recommendations for future enhancements or refinements to the deployed model based on ongoing monitoring and feedback.

Gantt Chart

Sprints	Days Worked on Each Sprint	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total Man-Days
Synthetic Data Generation	2	X	X													2
Data Cleaning	2			X	X											2
Feature Engineering	1					X										1
Model Training							X	X								2
Model Application and Iteration									X							1
Minimization of Bias and Overfitting										X						1
Test Models on Unseen Data or Validation Set											X	X				2
Final Model Selection and Conclusion													X			
Documentation														X	X	
Total Days		X	X	X	X	X	X	X	X	X	X	X				14