Sprint 4

Day 7 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 4 Day 7 Task 1

Documentation for Optimized Machine Learning Model

Overview

This documentation provides a detailed explanation of the steps taken to preprocess the data, train multiple machine learning models, evaluate their performance, and make predictions on new synthetic data. The goal is to identify the best-performing model and use it for future predictions.

Steps and Code

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
```

- **Scaling Numerical Columns**

  Numerical columns are scaled using StandardScaler.

```
# Scale numerical columns
def scale_numerical_columns(df, numeric_cols):
    # Initialize the StandardScaler
    scaler = StandardScaler()
    scaled_df =
pd.DataFrame(scaler.fit_transform(df[numeric_cols]),
columns=numeric_cols)

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

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
R-squared: 0.999999994464551

Model: Decision Tree
```

```
MSE: 9.702336836097705e-07
RMSE: 0.0009850044079138785
R-squared: 0.9999990225096674

Model: Random Forest
MSE: 3.860882999490959e-07
RMSE: 0.0006213600405152362
R-squared: 0.9999996110240377

Model: Gradient Boosting
MSE: 5.9453021697768875e-05
RMSE: 0.007710578557914372
R-squared: 0.999940102312531

Model: Support Vector Machine
MSE: 0.0030617833459808323
RMSE: 0.05533333836483983
R-squared: 0.9969153167203592

Best Model: Linear Regression
```

Predictions on New Data

- **Generating Synthetic Data**

  Create a synthetic dataset for testing.

```
import numpy as np

# Generate synthetic data
num_samples = 100  # Number of synthetic samples
new_data = pd.DataFrame({
    'Item Purchased': np.random.choice(df_cleaned['Item Purchased'].unique(), num_samples),
    'Transaction Mode': np.random.choice(df_cleaned['Transaction Mode'].unique(), num_samples),
    'Transaction Currency': np.random.choice(df_cleaned['Transaction Currency'].unique(), num_samples),
    'Unit Price': np.random.uniform(0, 100, num_samples),
    'Quantity': np.random.randint(1, 10, num_samples),
    'Tax': np.random.uniform(0, 20, num_samples),
    'Total Amount In INR': np.random.uniform(0, 1000, num_samples)
})
```

```python
# Preprocess the synthetic data
new_data_encoded = encode_categorical_columns(new_data,
categorical_columns)
new_data_preprocessed = scale_numerical_columns(new_data_encoded,
['Unit Price', 'Quantity', 'Tax', 'Total Amount In INR'])
# Make predictions using the best model
new_data_predictions = best_model.predict(new_data_preprocessed)

# Display the predictions
print(new_data_predictions)
```

Assessing Model Performance on Synthetic Data

Evaluate the model's performance using the synthetic data as a proxy for actual data.

```python
# Calculate evaluation metrics for the new data

mse_new_data = mean_squared_error(df_preprocessed['Transaction
Amount'][:100], new_data['Predicted Transaction Amount'])
rmse_new_data = np.sqrt(mse_new_data)
r2_new_data = r2_score(df_preprocessed['Transaction
Amount'][:100], new_data['Predicted Transaction Amount'])
# Display the evaluation metrics
print("Evaluation metrics for new data:")
print(f"  MSE: {mse_new_data}")
print(f"  RMSE: {rmse_new_data}")
print(f"  R-squared: {r2_new_data}")
```

```
Evaluation metrics for new data:
  MSE: 2.114131487869145
  RMSE: 1.4540053259424963
  R-squared: -1.330250012350632
```

Conclusion

This documentation outlines the process of using cross-validation to evaluate multiple machine learning models. Cross-validation provides a more robust measure of model performance by averaging the performance across multiple folds of the data, thereby reducing the variability due to random train-test splits. The best model is selected based on cross-validated RMSE scores and is then used to make predictions on new synthetic data. Finally, the performance of the model on synthetic data is assessed using standard evaluation metrics. This approach ensures a systematic and reliable way to build and evaluate a machine learning model.

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							
Minimization of Bias and Overfitting										X						
Test Models on Unseen Data or Validation Set																
Final Model Selection and Conclusion																
Documentation																
Total Days		X	X	X	X	X	X	X	X	X						14