Project Planning Phase-2

Technology Stack (Open source Frameworks)

Date	30-10-23
Team ID	592499
Project Name	Machine learning model for occupancy rates and demand in Hospitality industries

Team Members :

N Nitin : nitin.n2021@vitstudent.ac.in

Mekala Sujan : mekala.sujan2021@vitstudent.ac.in

Open source Frameworks

Open source solutions for building a machine learning model for predicting occupancy rates and demand in the hospitality industry, there are several open source frameworks and libraries that can be highly effective. Here's a selection of open source tools to consider:

1. Programming Languages:

- Python: As previously mentioned, Python is the primary language for machine learning and data science, and it has a rich ecosystem of open source libraries and tools.

2. Data Collection and Storage:

- SQL Databases:Open source relational databases such as PostgreSQL and MySQL can serve as the foundation for storing structured data.
- NoSQL Databases: Use open source NoSQL databases like MongoDB for semi-structured or unstructured data.

3. Data Processing and Analysis:

- Pandas: A powerful open source library for data manipulation and analysis in Python.
- Numpy: For numerical operations and array manipulation.
- Jupyter Notebooks: Ideal for interactive data exploration and visualization.

4. Machine Learning Frameworks:

- Scikit-Learn: An open source library for machine learning tasks, including regression and classification.
- XGBoost: An open source gradient boosting framework known for its effectiveness in predictive modeling.
- LightGBM: Another open source gradient boosting framework designed for efficiency and speed.
- CatBoost: A gradient boosting library that's open source and handles categorical features well.
- Prophet:An open source forecasting tool developed by Facebook for time series data.

5. Model Deployment:

- Flask or FastAPI: These open source Python web frameworks can be used to deploy machine learning models as REST APIs.
- Docker: Open source containerization for easier deployment and scaling.
- Kubernetes: For open source container orchestration and scaling if you have complex deployment needs.

6. Data Visualization:

- Matplotlib and Seaborn:Open source libraries for static data visualization.
- Plotly or Bokeh: For open source interactive and web-based visualization.

7. Monitoring and Logging:

- Prometheus and Grafana: Open source tools for monitoring model performance and system health.
- ELK Stack (Elasticsearch, Logstash, Kibana): Open source for log management and analysis.

8. Version Control:
- Git: Open source for code version control, collaboration, and tracking changes.

9. Automated Machine Learning (AutoML) Tools (Optional):
- While most AutoML tools are commercial, you can explore open source solutions like **Auto-Sklearn** for automated model selection and hyperparameter tuning.

10. Feedback and Communication:
- Slack or Mattermost:Open source alternatives for team communication and collaboration.
- Email or open source CRM systems: To communicate with clients or management.

11. Continuous Integration/Continuous Deployment (CI/CD):
- Use open source CI/CD tools like **Jenkins, GitLab CI, or Travis CI** for automated testing, deployment, and integration with your version control system.

12. Data Quality and ETL Tools (Optional):
- Open source tools like **Apache Nifi or Talend** can assist in data extraction, transformation, and loading (ETL).

13. A/B Testing (Optional):
- Open source A/B testing frameworks like **OpenAI Gym** can be used to conduct experiments.

Choosing open source frameworks and libraries can be cost-effective and provide flexibility. They also have strong communities and documentation for support and development. However, ensure that the open source tools you select align with your project's requirements and that you adhere to their respective licenses and terms.

R: R is another popular programming language for machine learning, and there are also a number of open source libraries available for R, such as caret, mlr, and tidymodels. These libraries can be used to train and deploy machine learning models for a variety of tasks, including predicting occupancy rates and demand.

Apache Spark: Apache Spark is a distributed computing framework that can be used to train and deploy machine learning models on large datasets. Spark can be used with both Python and R, and there are a number of open source machine learning libraries available for Spark, such as Spark MLlib and TensorFlow on Spark.

In addition to these general-purpose machine learning libraries, there are also a number of open source libraries that are specifically designed for the hospitality industry. For example, the following libraries can be used to build machine learning models for predicting occupancy rates and demand in the hospitality industry:

Hotel Booking Demand Forecasting: This Python library can be used to forecast hotel booking demand using a variety of machine learning algorithms.

Hotel Occupancy Rate Prediction: This Python library can be used to predict hotel occupancy rates using a variety of machine learning algorithms.

Tourism Demand Forecasting: This R library can be used to forecast tourism demand using a variety of machine learning algorithms.
Example: Building a machine learning model for predicting occupancy rates in Python

Python :

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Load the hotel booking data
hotel_booking_data = pd.read_csv('hotel_booking_data.csv')

# Prepare the data for training
X = hotel_booking_data[['historical_occupancy_rate', 'average_daily_rate', 'competitive_rates']]
y = hotel_booking_data['occupancy_rate']

# Split the data into training and test sets

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
```

```
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model's performance
from sklearn.metrics import mean_squared_error
print('Mean squared error:', mean_squared_error(y_test, y_pred))

# Plot the predicted occupancy rates against the actual occupancy rates
plt.scatter(y_test, y_pred)
plt.xlabel('Actual occupancy rate')
plt.ylabel('Predicted occupancy rate')
plt.title('Predicted vs. actual occupancy rates')
plt.show()
```

R :

```
# Load the hotel booking data
hotel_booking_data <- read.csv('hotel_booking_data.csv')

# Prepare the data for training
X <- hotel_booking_data[, c('historical_occupancy_rate', 'average_daily_rate', 'competitive_rates')]
y <- hotel_booking_data$occupancy_rate

# Split the data into training and test sets
library(caret)
set.seed(123)
trainIndex <- sample(1:nrow(X), size = nrow(X) * 0.75)
```

```
train <- X[trainIndex, ]
test <- X[-trainIndex, ]

# Create a linear regression model
library(mlr)
learner <- lrm()

# Train the model on the training data
model <- train(learner, train$X, train$y)

# Make predictions on the test data
y_pred <- predict(model, test$X)

# Evaluate the model's performance
library(Metrics)
mse <- mean_squared_error(y_pred, test$y)
print(mse)

# Plot the predicted occupancy rates against the actual occupancy rates
plot(y_pred, test$y, main = "Predicted vs. actual occupancy rates", xlab = "Actual occupancy rate", ylab = "Predicted
occupancy rate")
abline(0, 1, col = "gray")
```

Apache Spark code :

```
import numpy as np
import pandas as pd
from pyspark.sql import SparkSession
```

```
# Create a SparkSession
spark = SparkSession.builder.getOrCreate()

# Load the hotel booking data
hotel_booking_data = spark.read.csv('hotel_booking_data.csv', inferSchema=True)

# Prepare the data for training
X = hotel_booking_data.select('historical_occupancy_rate', 'average_daily_rate', 'competitive_rates')
y = hotel_booking_data.select('occupancy_rate')

# Split the data into training and test sets
from pyspark.ml.evaluation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# Create a linear regression model
from pyspark.ml.regression import LinearRegression

# Train the model on the training data
model = LinearRegression().fit(X_train)

# Make predictions on the test data
y_pred = model.transform(X_test).select('prediction')

# Evaluate the model's performance
from pyspark.ml.evaluation import RegressionEvaluator

# Create an evaluator
```

```
evaluator = RegressionEvaluator(metricName="mse")

# Calculate the mean squared error
mse = evaluator.evaluate(y_pred, y_test)

# Print the mean squared error
print('Mean squared error:', mse)

# Save the trained model
model.write().overwrite().save('occupancy_rate_prediction_model')
```