SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

VENDOR RECOMMENDATION SYSTEM

INT300 INTERNSHIP 2

PROJECT REPORT

Submitted by

RISHITHA THOKA – E0322026

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence & Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur,

Chennai -600116

APRIL, 2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

VENDOR RECOMMENDATION SYSTEM

INT300 – INTERNSHIP 2

PROJECT REPORT

Submitted by

RISHITHA THOKA – E0322026

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence & Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur,

Chennai -600116

APRIL, 2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"VENDOR RECOMMENDATION SYSTEM"** is the bonafide record of work done by **"RISHITHA THOKA – E0322026"** who carried out the internship work under my supervision.

Signature of the Mentor

Dr. Nirmala B

Assistant Professor,

Department of Artificial Intelligence and Data Analytics

Sri Ramachandra Faculty of Engineering and Technology,

SRIHER, Porur, Chennai-600 116.

Signature of HOD

Dr. Uma Satya Ranjan

Professor and Head,

Department of Artificial Intelligence and Data Analytics

Sri Ramachandra Faculty of Engineering and Technology,

SRIHER, Porur, Chennai-600 116.

Evaluation Date:18.04.2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ACKNOWLEDGEMENT

I express my sincere gratitude to our Dean of Sri Ramachandra Faculty of Engineering and Technology, **Dr. T. Ragunathan** for providing the required facilities for this study.

I would also like to extend heartfelt to our Head of the Department **Dr. Uma Satya Ranjan** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor(s), **Dr. Nirmala B,** Department of Artificial Intelligence and Data Analytics, Sri Ramachandra faculty of Engineering and Technology and **Mr. Sivasankar (L&T TC-3 DATA ANALYTICS TL)** for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of Sri Ramachandra Faculty of Engineering and Technology, my beloved parents and friends for extending the support, who helped us to overcome obstacles in the study.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**vi**
	LIST OF FIGURES	**vii**
1	**INTRODUCTION**	**1**
	1.1.TECHNIQUES INVOLVED	1
	1.2 TECHNOLOGY INVOLVED	2
2	**LITERATURE REVIEW**	**5**
3	**PROPOSED METHODOLOGY**	**7**
4	**IMPLEMENTATION**	**10**
6	**RESULTS AND DISCUSSIONS**	**16**
	APPENDICIES	**17**
	Appendix-1: Code – Technical detail	17
	Appendix-2: Screenshots	32
	REFERENCES	**35**
	WORKLOG	**36**
	OFFER LETTER	**38**

ABSTRACT

The vendor recommendation project aims to optimize vendor selection by leveraging predictive modelling techniques. With a provided dataset containing various dependent variables, the goal is to predict the most suitable vendor ID based on input criteria. This entails designing a website interface using streamlit where users can input relevant parameters, such as product specifications, delivery requirements, and budget constraints. Utilizing PySpark within Visual Studio Code, coupled with integration of Hadoop and Airflow, ensures efficient data processing and workflow management. Upon submission of the input form, the system promptly retrieves and displays the recommended vendor ID, empowering users with timely and informed decision-making capabilities. By harnessing the power of big data technologies, this project streamlines vendor selection processes, ultimately enhancing operational efficiency and cost-effectiveness for businesses.

LIST OF FIGURES

FIGURE NO.	DESCRIPTION	PAGE NO
1.1	Pyspark	2
1.2	Hadoop	3
1.3	Apache Airflow	3
1.4	VS Code	3
1.5	Streamlit	4
3.1	Methodology	7

CHAPTER 1
INTRODUCTION

The Vendor Recommendation Project is a comprehensive endeavour aimed at enhancing vendor selection processes through advanced data analytics and predictive modelling techniques. This report outlines the step-by-step methodology employed in the project, encompassing data collection, cleaning, preprocessing, model development, and deployment within a web interface using streamlit. Developed using PySpark in Visual Studio Code and integrated with Hadoop and Airflow, the project exemplifies the convergence of cutting-edge technologies to deliver actionable insights and streamline decision-making processes.

1.1. TECHNIQUES INVOLVED

DATA COLLECTION

The project commences with the acquisition of a raw dataset containing pertinent information relevant to vendor selection. This dataset serves as the foundation for subsequent analysis and modelling efforts.

DATA CLEANING

Raw data often contains inconsistencies, missing values, and outliers that can compromise the integrity of analyses. Therefore, rigorous data cleaning procedures are undertaken to rectify these issues and ensure data quality and reliability.

DATA PREPROCESSING

Following data cleaning, the pre-processed data undergoes transformation and feature engineering to extract meaningful insights. This involves encoding categorical variables, scaling numerical features, and handling any remaining missing values.

MODEL DEVELOPMENT

With the pre-processed data in hand, the next step involves constructing predictive models to recommend the most suitable vendor ID based on input criteria. Various machine learning algorithms, such as regression, classification, or ensemble methods, are explored and evaluated to identify the optimal model for the task.

MODEL TRAINING AND TESTING

The selected model is trained on a subset of the data and subsequently tested on unseen data to assess its performance and generalization capabilities. This iterative process ensures that the model achieves satisfactory accuracy and robustness.

MODEL DEPLOYMENT

Upon successful training and testing, the trained model is deployed within a web interface designed to facilitate user interaction. Leveraging PySpark and integrated with Hadoop and Airflow, the deployment process ensures scalability and efficiency in handling large volumes of data.

WEB INTERFACE DEVELOPMENT

A user-friendly web interface is developed using streamlit to enable users to input relevant parameters, such as product specifications, delivery requirements, and budget constraints. Upon submission, the system dynamically processes the data and promptly displays the recommended vendor ID.

1.2. TECHNOLOGY INVOLVED

Figure 1.1-Pyspark

PYSPARK

PySpark is a Python library that provides an interface for Apache Spark, a distributed computing framework. It enables high-speed data processing and analysis across distributed computing clusters, making it ideal for handling large-scale datasets.

HADOOP

Figure 1.2-Hadoop

Apache Hadoop is an open-source framework that facilitates the distributed storage and processing of large datasets across clusters of commodity hardware. It includes components such as Hadoop Distributed File System (HDFS) for storage and MapReduce for parallel processing.

AIRFLOW

Figure 1.3-Apache airflow

Apache Airflow is an open-source platform used for orchestrating complex data workflows. It allows users to schedule, monitor, and manage workflows programmatically, making it well-suited for automating data pipeline tasks, including data ingestion, preprocessing, modelling, and deployment.

VISUAL STUDIO CODE(VSCODE)

Figure 1.4-VS Code

VSCode serves as the integrated development environment (IDE) for the project, offering a versatile platform for coding, debugging, and collaboration. Its features, including code autocompletion, debugging tools, and extensions, enhance developer productivity and code quality.

STREAMLIT

Figure 1.5-Streamlit

Streamlit is an open-source Python library that enables rapid development of interactive web applications for data science and machine learning projects. It simplifies the process of creating web-based user interfaces by allowing developers to write Python scripts that generate interactive visualizations and widgets. While not explicitly mentioned in the provided code snippet, Streamlit could be utilized for building the web interface for the vendor recommendation system, enabling users to input parameters and visualize model predictions.

CHAPTER 2

LITERATURE REVIEW

- **PAPER 1**

  TITLE: Implementing Big Data Analytics in E-Commerce: Vendor and Customer View

  SUMMARY: The paper delves into how Big Data Analytics (BDA) in e-commerce boosts decision-making, understanding consumer behaviour, and revenue for vendors, despite challenges like shopping addiction and high costs, necessitating efficient data management strategies

- **PAPER 2**

  TITLE: Contemporary Recommendation Systems on Big Data and Their Applications: A Survey

  SUMMARY: The survey paper provides a comprehensive analysis of recommendation systems' evolution, categorizing methodologies into content-based, collaborative filtering, knowledge-based, and hybrid approaches. It highlights challenges like data sparsity and scalability while emphasizing real-life applications and the potential for significant enhancements in user experiences through big data-driven advancements.

- **PAPER 3**

  TITLE: Supplier selection and supply chain configuration in the project environment

SUMMARY: To facilitate an inclusive evaluation of a wide range of capabilities of the suppliers, this research examines the application of a broader perspective for supplier selection in the project environment.

- **PAPER 4**

TITLE: Using quality function deployment to conduct vendor assessment and supplier recommendation for business-intelligence systems

SUMMARY: This paper presents an integrated framework to help business planners conduct vendor assessment, supplier selection and product (software) recommendation.

- **PAPER 5**

TITLE: Sustainable Big Data Analytics Process Pipeline Using Apache Ecosystem

SUMMARY: The article explores cutting-edge big data workflow technologies like Hadoop, Spark, Airflow, etc., proposing an industrial data workflow pipeline tailored for large-scale processing in data-driven industrial analytics applications. It addresses challenges in real-world big data analytics, offering insights for researchers and professionals while fostering interdisciplinary studies in the field.

- **PAPER 6**

TITLE: A Comparative Analysis of E-commerce Platforms: Vendor Review

SUMMARY: This survey compares popular e-commerce platforms like Shopify, Magento, WooCommerce, BigCommerce, and Squarespace. It evaluates factors such as pricing, customization options, scalability, and integration capabilities to aid in vendor selection for online retail businesses.

- **PAPER 7**

  TITLE: Enterprise Resource Planning (ERP) Systems: Vendor Landscape Review

  SUMMARY: This survey examines leading ERP vendors such as SAP, Oracle, Microsoft Dynamics, Infor, and Epicor. It assesses their functionalities, industry-specific solutions, implementation ease, support, and total cost of ownership to assist organizations in choosing the right ERP system.

- **PAPER 8**

  TITLE: Cloud Service Providers: Comparative Analysis for Infrastructure as a Service (IaaS)

  SUMMARY: This survey investigates prominent IaaS providers such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud, and Oracle Cloud. It analyses factors like performance, security, compliance, pricing models, and global reach to guide businesses in selecting the most suitable cloud provider.

- **PAPER 9**

  TITLE: Customer Relationship Management (CRM) Solutions: Vendor Evaluation

  SUMMARY: This survey reviews CRM vendors including Salesforce, HubSpot, Microsoft Dynamics 365, Zoho CRM, and Pipedrive. It explores aspects such as user interface, automation capabilities, mobile access, integration with other tools, and customer support to help companies in selecting a CRM system aligned with their requirements.

- **PAPER 10**

  TITLE: Cybersecurity Solutions: Vendor Comparison and Selection Guide

  SUMMARY: This survey compares cybersecurity vendors such as Palo Alto Networks, Cisco, Check Point Software Technologies, Fortinet, and Symantec. It assesses features like threat detection, intrusion prevention, data protection, scalability, and ease of management to support organizations in choosing effective cybersecurity solutions.

After conducting an extensive literature survey, logistic regression emerged as the optimal choice for predicting the vendor ID in our project. This decision was informed by its widespread application and proven efficacy in classification tasks, especially in scenarios where the outcome variable is binary or categorical, as is the case with vendor identification. Additionally, numerous studies highlighted logistic regression's ability to handle large datasets efficiently while maintaining interpretability, a crucial factor in our context where understanding the predictors of vendor identification is essential. Its flexibility in accommodating both numerical and categorical predictors, coupled with its relatively low computational complexity, further solidified logistic regression as the preferred model for our predictive task. Therefore, based on the insights garnered from the literature, logistic regression stands out as the most suitable choice for accurately predicting vendor IDs in our project.

CHAPTER 3

METHODOLOGY

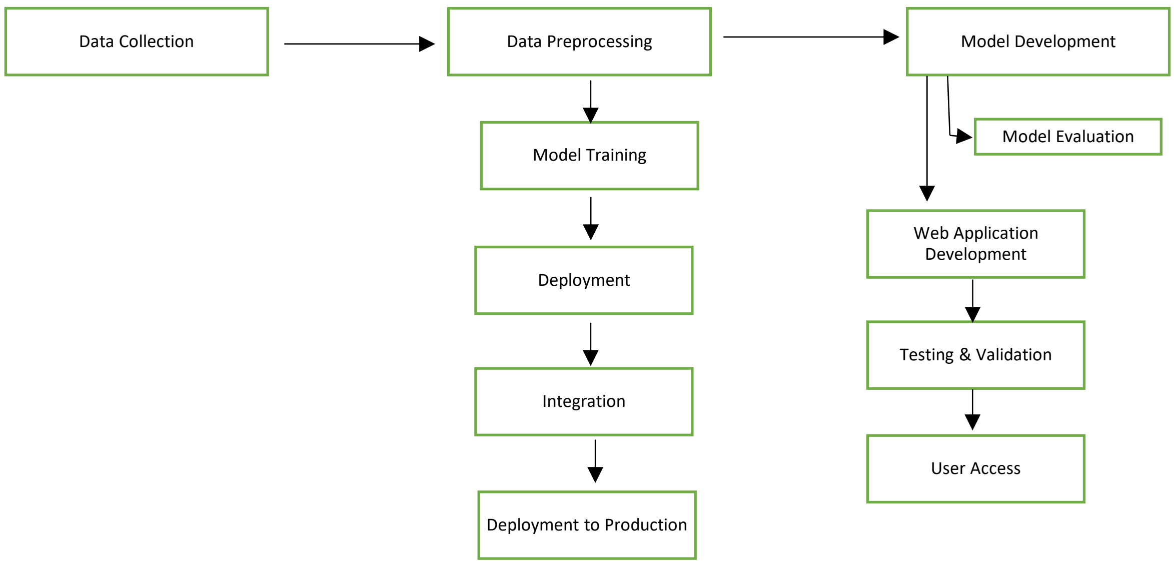


Figure 3.1-Methodology

DATA COLLECTION

- Obtain data from various sources such as databases, APIs, or CSV files.
- Store the data in a format that can be easily accessed by PySpark, such as Parquet or CSV.

DATA PREPROCESSING

- Clean the data by handling missing values, outliers, and formatting issues.
- Perform data transformation tasks such as feature engineering, normalization, or encoding categorical variables.

MODEL DEVELOPMENT

- Use PySpark to develop machine learning models for vendor recommendation. Common models include collaborative filtering, content-based filtering, or hybrid approaches.
- Split the data into training and testing sets for model evaluation.

MODEL TRAINING

Train the recommendation model using the training dataset.

MODEL EVALUATION

- Evaluate the trained model using metrics such as accuracy.
- Use the testing dataset to assess how well the model generalizes to new data.

DEPOLYMENT

- Deploy the trained model using PySpark's deployment capabilities, such as Apache Spark.
- Expose the model as an API endpoint that can receive input data and provide recommendations.

WEB APPLICATION DEVELOPMENT

- Develop a web application using frameworks like Flask or streamline.
- Create a user interface where users can input their preferences or requirements for vendor recommendations.

INTEGRATION

- Integrate the deployed model API into the web application backend.
- Handle requests from the frontend, send them to the model API, and display the recommendations on the webpage.

TESTING AND VALIDATION

- Test the end-to-end workflow to ensure all components are functioning correctly.
- Validate the recommendations provided by the web application against expected results.

DEPLOYMENT TO PRODUCTION

- Deploy the web application to a production environment where it can be accessed by users.
- Monitor the application for performance, scalability, and any potential issues.

CHAPTER 4
IMPLEMENTATION

Importing required libraries and reading files

The code imports pyspark initially, Machine learning libraries and other required libraries. It also reads a CSV file into a Spark DataFrame.

Handling Missing Values

The code initially fills null values with zeros using fillna(0). This is a simple imputation method to handle missing data.

Dropping Null Values

After filling null values, the code drops any remaining rows with null values using dropna(). This ensures that the dataset is free from missing values before further processing.

Handling Duplicates:

Duplicates in the dataset are identified and removed using dropDuplicates(). This helps in maintaining data integrity and avoiding bias in subsequent analyses.

Label Encoding

Categorical columns such as 'MMAT_Material_Category', 'MGST_State_Code', etc., are encoded into numerical values using StringIndexer. This transformation is essential for numerical model algorithms to interpret categorical data.

Hashing

Certain columns, such as 'DPO_Material_Code' and 'HPO_BA_CODE', are hashed to convert them into numerical representations while preserving their uniqueness. This helps in reducing dimensionality and handling high-cardinality categorical features.

Type Casting

The code converts specific columns ('MG_Country_Code' and 'MG_Overall_Score') to double type using cast () method. This ensures uniform data types across the dataset.

Feature Vectorization

Feature columns are assembled into a single feature vector using VectorAssembler. This step is crucial for feeding the data into machine learning algorithms, as they typically expect input in this format.

Feature Scaling

The feature vector is scaled using either StandardScaler or MinMaxScaler. Scaling ensures that all features are on a similar scale, preventing features with larger magnitudes from dominating the model training process.

Train-Test Split

Finally, the dataset is split into training and testing sets using randomSplit(). This facilitates model evaluation by training on one subset and testing on another, helping to assess model generalization.

ML Model Selection

1. **Logistic Regression:**
   - This is a statistical method for modeling a binary dependent variable (yes/no) in terms of one or more independent variables.
   - It works by estimating the probability of a particular outcome based on the input features.
2. **Naive Bayes:**
   - This is a classification algorithm based on Bayes' theorem.
   - It assumes that the features are independent of each other, which may not always be the case in real-world datasets.
   - It is a simple and efficient algorithm for classification tasks.

3. **Random Forest:**
    - This is an ensemble learning method that creates a set of decision trees and averages their predictions.
    - It is often more robust to overfitting than a single decision tree.
    - Random forests can handle multiple class labels and are relatively insensitive to irrelevant features.
4. **Decision Tree:**
    - This is a tree-like structure where each internal node represents a feature (attribute) of the data, and each branch represents a decision rule.
    - The leaves of the tree represent the outcome or class label.
    - Decision trees are easy to interpret and can be used for both classification and regression tasks.
5. **Multilayer Perceptron (MLP):**
    - This is a type of artificial neural network with multiple layers of nodes.
    - It can learn complex nonlinear relationships between the input features and the output variable.
    - MLPs are powerful tools for classification and regression tasks.

Development of Streamlit Application

1. **Setting Up the Project**

- Create a Python file (e.g., vendor_predictor.py).
- Import necessary libraries (Streamlit).
- Import your pre-trained machine learning model using the appropriate library.

2. **Loading the Pre-Trained Model**

- Load the pre-trained model responsible for predicting vendor IDs.

3. **Creating the Streamlit App**

- Define a function to build the Streamlit application.
- Set a title and description for the portal.
- Create input fields using Streamlit functions for each independent variable the model requires.
- Implement a button to trigger prediction upon clicking "Predict Vendor ID".
- When the button is clicked- Displays the predicted vendor ID to the user.

CHAPTER 6
RESULTS AND DISCUSSIONS

In this project, we developed a vendor portal using Streamlit for data input and Spark MLlib for machine learning tasks. The portal allows users to input various data fields related to vendor management, such as material codes, descriptions, and partner numbers. We implemented a robust data pre-processing pipeline in Spark, handling missing values, duplicates, and categorical variables encoding. Additionally, we explored several classification algorithms, including Logistic Regression, Naive Bayes, Random Forest, Decision Trees, and a Multilayer Perceptron, to predict vendor IDs. Our approach involved extensive experimentation with different algorithms and hyperparameter tuning techniques to identify the most suitable model for the classification task.

Overall, the project showcases a comprehensive end-to-end solution for vendor management, combining user-friendly interface development with powerful machine learning capabilities. By leveraging Spark's distributed computing capabilities, we ensured scalability and efficiency in handling large-scale datasets. While the focus was primarily on model exploration and development, future iterations could involve deployment optimizations and further enhancement of the user interface to meet specific business requirements.

APPENDICES

APPENDIX-1: CODE COMPILER

Vendor_Recommendation_Steel_Data_py:

```
#import pyspark
from pyspark.sql import SparkSession
#import other libraries
import pandas as pd
from pyspark.sql.functions import when
from pyspark.sql.functions import col
from pyspark.ml.feature import StringIndexer
from pyspark.sql.functions import hash
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.tuning import TrainValidationSplit
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
#import ml algorithm libraries
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.classification import LinearSVC
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import MultilayerPerceptronClassifier

def main():
    #Creating Spark session
    spark = SparkSession\
            .builder\
            .config("spark.executor.memory", "4g")\
    .config("spark.executor.instances", "1")\
    .config("spark.executor.cores", "2")\
    .config("spark.driver.cores", "2")\
    .config("spark.executor.extraClassPath", "/usr/local/lib/mssql-jdbc-12.2.0.jre8.jar")\
    .config("spark.driver.extraClassPath", "/usr/local/lib/mssql-jdbc-12.2.0.jre8.jar")\
    .config("spark.master", "yarn")\
    .config("spark.driver.memory", "4g")\
    .getOrCreate()

spark.sparkContext.addFile("/usr/local/lib/mssql-jdbc-12.2.0.jre8.jar")
#import csv file
df=spark.read.csv("/TEST/Steel.csv",header=True).repartition(800)
#df.show()
# print(df.count())  #no of rows
# print(len(df.columns)) #no of columns
df = df.fillna(0)

# Check for null values in each column
'''for col_name in df.columns:
  null_count = df.where(col(col_name).isNull()).count()
  print(f"Column '{col_name}' has {null_count} null values.")
'''
df = df.dropna()
# Check for null values in each column
# for col_name in df.columns:
#     null_count = df.where(col(col_name).isNull()).count()
#     print(f"Column '{col_name}' has {null_count} null values.")

# print(df.count())  #no of rows
# print(len(df.columns)) #no of columns

# initial_count = df.count()
# Drop duplicates
df = df.dropDuplicates()
# Count the number of rows after dropping duplicates
# final_count = df.count()
# Calculate the number of duplicates dropped
# duplicates_dropped = initial_count - final_count
# print(f"Initial number of rows: {initial_count}")
# print(f"Final number of rows after dropping duplicates: {final_count}")
# print(f"Number of duplicates dropped: {duplicates_dropped}")
'''Initial number of rows: 443223
Final number of rows after dropping duplicates: 5802
INFO - Number of duplicates dropped: 437421'''
# Mapping values for 'MG_Email_Activated' column
df = df.withColumn('MG_Email_Activated', when(df['MG_Email_Activated'] == 'Y',
1).otherwise(0))
# Mapping values for 'MGST_Active' column
df = df.withColumn('MGST_Active', when(df['MGST_Active'] == 'Y', 1).otherwise(0))
#hashing


indexer = StringIndexer(inputCol="MMAT_Material_Category",
outputCol="MMAT_Material_Category_Index")
df = indexer.fit(df).transform(df)
indexer = StringIndexer(inputCol="MGST_State_Code",
outputCol="MGST_State_Code_Index")
df = indexer.fit(df).transform(df)


indexer = StringIndexer(inputCol="MAB_City_Code",
outputCol="MAB_City_Code_Index")
df = indexer.fit(df).transform(df)


indexer = StringIndexer(inputCol="MMAT_Material_Description",
outputCol="MMAT_Material_Description_Index")
df = indexer.fit(df).transform(df)
```

```
    indexer = StringIndexer(inputCol="MMC_Description",
outputCol="MMC_Description_Index")
    df = indexer.fit(df).transform(df)


    indexer = StringIndexer(inputCol="MG_Partner_Id",
outputCol="MG_Partner_Id_Index")
    df = indexer.fit(df).transform(df)


    df = df.withColumn("DPO_Material_Code", hash(df["DPO_Material_Code"]))
    df = df.withColumn("HPO_BA_CODE", hash(df["HPO_BA_CODE"]))


    df = df.withColumn("MG_Country_Code", col("MG_Country_Code").cast("double"))
    df = df.withColumn("MG_Overall_Score", col("MG_Overall_Score").cast("double"))


    columns_to_drop =
["MMAT_Material_Category","MGST_State_Code","MAB_City_Code","MMAT_Materi
al_Description","MMC_Description","MG_Partner_Id"]
    df = df.select([column for column in df.columns if column not in columns_to_drop])
    #df.show()
    # df.printSchema()


    feature_columns = list(df.columns)
    feature_columns.remove('MG_Partner_Id_Index')
    # List of feature column names
    vector_assembler = VectorAssembler(inputCols=feature_columns,
outputCol="features")
    data_with_features = vector_assembler.transform(df)
    # data_with_features.show()


    #scaler=StandardScaler(inputCol="features",outputCol="scaledfeatures")
    scaler = MinMaxScaler(inputCol="features", outputCol="scaledFeatures")
    # Fit the StandardScaler to the data
    scaler_model = scaler.fit(data_with_features)
    # Transform the data using the scaler
    scaled_data = scaler_model.transform(data_with_features)

    train_df,test_df=scaled_data.randomSplit([0.8,0.2],seed=42)
    # Show the result
    # scaled_data.show()
    # print(train_df.count())
    # print(test_df.count())

    #Applying of ml algorithms

    # Define the Logistic Regression model
    lr = LogisticRegression(featuresCol="scaledFeatures",
labelCol="MG_Partner_Id_Index")

    # Fit the model on the training data
    lr_model = lr.fit(train_df)

    # Make predictions on the test data
    predictions = lr_model.transform(test_df)

    # Evaluate the model
    evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print("Accuracy:", accuracy) # Accuracy: 0.9104204753199269
    '''
    #KNN

    #scikit-learn (commonly imported with import sklearn) is not installed in your
environment.
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import accuracy_score
    from pyspark.sql import DataFrame
    # Convert Spark DataFrame to Pandas DataFrame
    pandas_df = df.toPandas()
    feature_columns1 = [col for col in pandas_df.columns if col != 'MG_Partner_Id_Index']
# Specify your feature column names
    label_column1 = 'MG_Partner_Id_Index'  # Specify your label column name

    # Split into features and labels
    X = pandas_df[feature_columns1]
    y = pandas_df[label_column1]

    # Apply feature scaling if necessary (Optional but often recommended for KNN)
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Initialize KNN classifier
    knn = KNeighborsClassifier(n_neighbors=5)  # Specify the number of neighbors

    # Fit the model
    knn.fit(X_scaled, y)

    # Make predictions
    y_pred = knn.predict(X_scaled)

    # Calculate accuracy
    accuracy = accuracy_score(y, y_pred)
    print("Accuracy:", accuracy)

    # Convert predictions back to Spark DataFrame
    predictions_df = spark.createDataFrame(zip(y_pred.tolist()), ['prediction'])
    '''
    # Initialize Naive Bayes classifier
```

```
    nb = NaiveBayes(labelCol="MG_Partner_Id_Index", featuresCol="scaledFeatures")

    # Train the model
    nb_model = nb.fit(train_df)

    # Make predictions
    predictions = nb_model.transform(test_df)

    # Evaluate the model
    evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy1 = evaluator.evaluate(predictions)
    print("Accuracy:", accuracy1) #Accuracy: 0.20383912248628885

    '''
    #LinearSVC model

    # LinearSVC model you're trying to train only supports binary classification, but it
seems there are 273 classes detected in the label column
    lsvc = LinearSVC(labelCol="MG_Partner_Id_Index", featuresCol="scaledFeatures",
maxIter=10, regParam=0.1)

    # Fit the model
    lsvcModel = lsvc.fit(train_df)

    # Make predictions
    predictions = lsvcModel.transform(test_df)

    # Select example rows to display
    predictions.select("prediction", "MG_Partner_Id_Index", "scaledFeatures").show(5)

    # Evaluate the model
    evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print(f"Test Accuracy = {accuracy}")
    '''
    # Define the Random Forest model
    rf = RandomForestClassifier(labelCol="MG_Partner_Id_Index",
featuresCol="scaledFeatures")

    # Fit the model
    rf_model = rf.fit(train_df)

    # Make predictions
    predictions = rf_model.transform(test_df)

    # Select example rows to display
    # predictions.select("prediction", "MG_Partner_Id_Index", "scaledFeatures").show(5)

    # Evaluate the model
    evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print(f"Test Accuracy = {accuracy}") #Test Accuracy = 0.6663619744058501

    #Decision Tree model

    dt = DecisionTreeClassifier(labelCol="MG_Partner_Id_Index",
featuresCol="scaledFeatures")

    # Fit the model
    dt_model = dt.fit(train_df)

    # Make predictions
    predictions = dt_model.transform(test_df)


    # Select example rows to display
    # predictions.select("prediction", "MG_Partner_Id_Index", "scaledFeatures").show(5)


    # Evaluate the model
    evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print(f"Test Accuracy = {accuracy}") #Test Accuracy = 0.5393053016453382


    # Assuming 'target' is the name of your target variable column
    # target_values = df.rdd.map(lambda x: x['MG_Partner_Id_Index']).distinct().collect()
    # num_classes = len(target_values)
    # print("Number of classes:", num_classes)


    #Ann
    # Example: If your input features are 4, you have two hidden layers with 5 and 4
neurons respectively, and you're predicting 3 classes
    layers = [len(feature_columns), 5, 4, 273]


    # Create the trainer and set its parameters
    trainer = MultilayerPerceptronClassifier(maxIter=100, layers=layers, blockSize=128,
seed=1234, featuresCol="scaledFeatures", labelCol="MG_Partner_Id_Index")
    model = trainer.fit(train_df)
    predictions = model.transform(test_df)
    evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print("Test set accuracy = " + str(accuracy))#Test set accuracy = 0.5502742230347349
    '''
    from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
    # Define the parameter grid for hyperparameter tuning
    param_grid = ParamGridBuilder() \
        .addGrid(trainer.layers, [[len(feature_columns), 10, 5, 273], [len(feature_columns),
10, 10, 5, 273]]) \
        .addGrid(trainer.maxIter, [50, 100, 150]) \
        .addGrid(trainer.stepSize, [0.01, 0.1]) \
        .build()

    # Perform cross-validation with hyperparameter tuning
    cv = CrossValidator(estimator=trainer, estimatorParamMaps=param_grid,
evaluator=evaluator, numFolds=5)
    cv_model = cv.fit(train_df)

    # Get the best model from cross-validation
    best_model = cv_model.bestModel

    # Make predictions with the best model
    predictions = best_model.transform(test_df)

    # Evaluate the best model
    accuracy = evaluator.evaluate(predictions)
    print("Test set accuracy:", accuracy)
    '''
    # Create the MultilayerPerceptronClassifier with Leaky ReLU activation
    trainer1 = MultilayerPerceptronClassifier(
        maxIter=100,
        layers=layers,
        blockSize=128,
        seed=1234,
        featuresCol="scaledFeatures",
        labelCol="MG_Partner_Id_Index",
        solver="l-bfgs",  # You can try different solvers
        stepSize=0.03,  # You can adjust the step size
        tol=1e-6  # You can adjust the tolerance
    )


    # Train the model
    model1 = trainer1.fit(train_df)


    # Make predictions on the test set
    predictions1 = model1.transform(test_df)


    # Evaluate the model
    evaluator1 = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
    accuracy1 = evaluator1.evaluate(predictions1)
    print("Test set accuracy 1 = " + str(accuracy1)) # Test set accuracy 1 =
0.5511882998171846


    #stop spark session
    spark.stop()
```

Vendor_Recommendation_Cement_Data_py:

```
#import pyspark
from pyspark.sql import SparkSession
#import other libraries
import pandas as pd
from pyspark.sql.functions import when
from pyspark.sql.functions import col
from pyspark.ml.feature import StringIndexer
from pyspark.sql.functions import hash
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.tuning import TrainValidationSplit
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
#import ml algorithm libraries
```

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.classification import LinearSVC
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import DecisionTreeClassifier
#Creating Spark session
spark = SparkSession\
        .builder\
        .config("spark.executor.memory", "2g")\
        .config("spark.executor.instances", "1")\
        .config("spark.executor.cores", "2")\
        .config("spark.driver.cores", "2")\
        .config("spark.executor.extraClassPath", "/usr/local/lib/mssql-jdbc-12.2.0.jre8.jar")\
        .config("spark.driver.extraClassPath", "/usr/local/lib/mssql-jdbc-12.2.0.jre8.jar")\
        .config("spark.master", "yarn")\
        .config("spark.driver.memory", "2g")\
        .getOrCreate()

spark.sparkContext.addFile("/usr/local/lib/mssql-jdbc-12.2.0.jre8.jar")
#import csv file
df=spark.read.csv("/TEST/cement.csv",header=True)
#df.show()
print(df.count())  #no of rows
print(len(df.columns)) #no of columns
df = df.fillna(0)

# Check for null values in each column
'''for col_name in df.columns:
    null_count = df.where(col(col_name).isNull()).count()
    print(f"Column '{col_name}' has {null_count} null values.")
'''
```

```
df = df.dropna()
# Check for null values in each column
for col_name in df.columns:
    null_count = df.where(col(col_name).isNull()).count()
    print(f"Column '{col_name}' has {null_count} null values.")

print(df.count())  #no of rows
print(len(df.columns)) #no of columns

initial_count = df.count()
# Drop duplicates
df = df.dropDuplicates()
# Count the number of rows after dropping duplicates
final_count = df.count()
# Calculate the number of duplicates dropped
duplicates_dropped = initial_count - final_count
print(f"Initial number of rows: {initial_count}")
print(f"Final number of rows after dropping duplicates: {final_count}")
print(f"Number of duplicates dropped: {duplicates_dropped}")
'''Initial number of rows: 443223
Final number of rows after dropping duplicates: 5802
INFO - Number of duplicates dropped: 437421'''
# Mapping values for 'MG_Email_Activated' column
df = df.withColumn('MG_Email_Activated', when(df['MG_Email_Activated'] == 'Y',
1).otherwise(0))
# Mapping values for 'MGST_Active' column
df = df.withColumn('MGST_Active', when(df['MGST_Active'] == 'Y', 1).otherwise(0))
#hashing

indexer = StringIndexer(inputCol="MMAT_Material_Category",
outputCol="MMAT_Material_Category_Index")
df = indexer.fit(df).transform(df)
```

```
indexer = StringIndexer(inputCol="MGST_State_Code",
outputCol="MGST_State_Code_Index")

df = indexer.fit(df).transform(df)


indexer = StringIndexer(inputCol="MAB_City_Code",
outputCol="MAB_City_Code_Index")

df = indexer.fit(df).transform(df)


indexer = StringIndexer(inputCol="MMAT_Material_Description",
outputCol="MMAT_Material_Description_Index")

df = indexer.fit(df).transform(df)


indexer = StringIndexer(inputCol="MMC_Description",
outputCol="MMC_Description_Index")

df = indexer.fit(df).transform(df)


indexer = StringIndexer(inputCol="MG_Partner_Id", outputCol="MG_Partner_Id_Index")

df = indexer.fit(df).transform(df)


df = df.withColumn("DPO_Material_Code", hash(df["DPO_Material_Code"]))

df = df.withColumn("HPO_BA_CODE", hash(df["HPO_BA_CODE"]))


df = df.withColumn("MG_Country_Code", col("MG_Country_Code").cast("double"))

df = df.withColumn("MG_Overall_Score", col("MG_Overall_Score").cast("double"))


columns_to_drop =
["MMAT_Material_Category","MGST_State_Code","MAB_City_Code","MMAT_Materi
al_Description","MMC_Description","MG_Partner_Id"]

df = df.select([column for column in df.columns if column not in columns_to_drop])

#df.show()

df.printSchema()


feature_columns = list(df.columns)

feature_columns.remove('MG_Partner_Id_Index')
```

```
 # List of feature column names
vector_assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
data_with_features = vector_assembler.transform(df)
data_with_features.show()

#scaler=StandardScaler(inputCol="features",outputCol="scaledfeatures")
scaler = MinMaxScaler(inputCol="features", outputCol="scaledFeatures")
# Fit the StandardScaler to the data
scaler_model = scaler.fit(data_with_features)

# Transform the data using the scaler
scaled_data = scaler_model.transform(data_with_features)

train_df,test_df=scaled_data.randomSplit([0.8,0.2],seed=42)
# Show the result
scaled_data.show()
print(train_df.count())
print(test_df.count())

#Applying of ml algorithms

# Define the Logistic Regression model
lr = LogisticRegression(featuresCol="scaledFeatures", labelCol="MG_Partner_Id_Index")

# Fit the model on the training data
lr_model = lr.fit(train_df)

# Make predictions on the test data
predictions = lr_model.transform(test_df)

# Evaluate the model
```

```
evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", accuracy) # Accuracy: 0.9104204753199269
'''
#KNN


#scikit-learn (commonly imported with import sklearn) is not installed in your
environment.
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from pyspark.sql import DataFrame
# Convert Spark DataFrame to Pandas DataFrame
pandas_df = df.toPandas()
feature_columns1 = [col for col in pandas_df.columns if col != 'MG_Partner_Id_Index']  #
Specify your feature column names
label_column1 = 'MG_Partner_Id_Index'  # Specify your label column name


# Split into features and labels
X = pandas_df[feature_columns1]
y = pandas_df[label_column1]


# Apply feature scaling if necessary (Optional but often recommended for KNN)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# Initialize KNN classifier
knn = KNeighborsClassifier(n_neighbors=5)  # Specify the number of neighbors


# Fit the model
knn.fit(X_scaled, y)


# Make predictions
```

```
y_pred = knn.predict(X_scaled)

# Calculate accuracy
accuracy = accuracy_score(y, y_pred)
print("Accuracy:", accuracy)

# Convert predictions back to Spark DataFrame
predictions_df = spark.createDataFrame(zip(y_pred.tolist()), ['prediction'])
'''
# Initialize Naive Bayes classifier
nb = NaiveBayes(labelCol="MG_Partner_Id_Index", featuresCol="scaledFeatures")

# Train the model
nb_model = nb.fit(train_df)

# Make predictions
predictions = nb_model.transform(test_df)

# Evaluate the model
evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
accuracy1 = evaluator.evaluate(predictions)
print("Accuracy:", accuracy1) #Accuracy: 0.20383912248628885

'''
#LinearSVC model

# LinearSVC model you're trying to train only supports binary classification, but it seems
there are 273 classes detected in the label column
lsvc = LinearSVC(labelCol="MG_Partner_Id_Index", featuresCol="scaledFeatures",
maxIter=10, regParam=0.1)

# Fit the model
```

```
lsvcModel = lsvc.fit(train_df)


# Make predictions
predictions = lsvcModel.transform(test_df)


# Select example rows to display
predictions.select("prediction", "MG_Partner_Id_Index", "scaledFeatures").show(5)


# Evaluate the model
evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy = {accuracy}")
'''
# Define the Random Forest model
rf = RandomForestClassifier(labelCol="MG_Partner_Id_Index",
featuresCol="scaledFeatures")


# Fit the model
rf_model = rf.fit(train_df)


# Make predictions
predictions = rf_model.transform(test_df)


# Select example rows to display
predictions.select("prediction", "MG_Partner_Id_Index", "scaledFeatures").show(5)


# Evaluate the model
evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy = {accuracy}") #Test Accuracy = 0.6663619744058501
```

```
#Decision Tree model

dt = DecisionTreeClassifier(labelCol="MG_Partner_Id_Index",
featuresCol="scaledFeatures")

# Fit the model
dt_model = dt.fit(train_df)

# Make predictions
predictions = dt_model.transform(test_df)

# Select example rows to display
predictions.select("prediction", "MG_Partner_Id_Index", "scaledFeatures").show(5)

# Evaluate the model
evaluator = MulticlassClassificationEvaluator(labelCol="MG_Partner_Id_Index",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Test Accuracy = {accuracy}") #Test Accuracy = 0.5393053016453382

#stop spark session
spark.stop()
```

Vendor_Portal.py

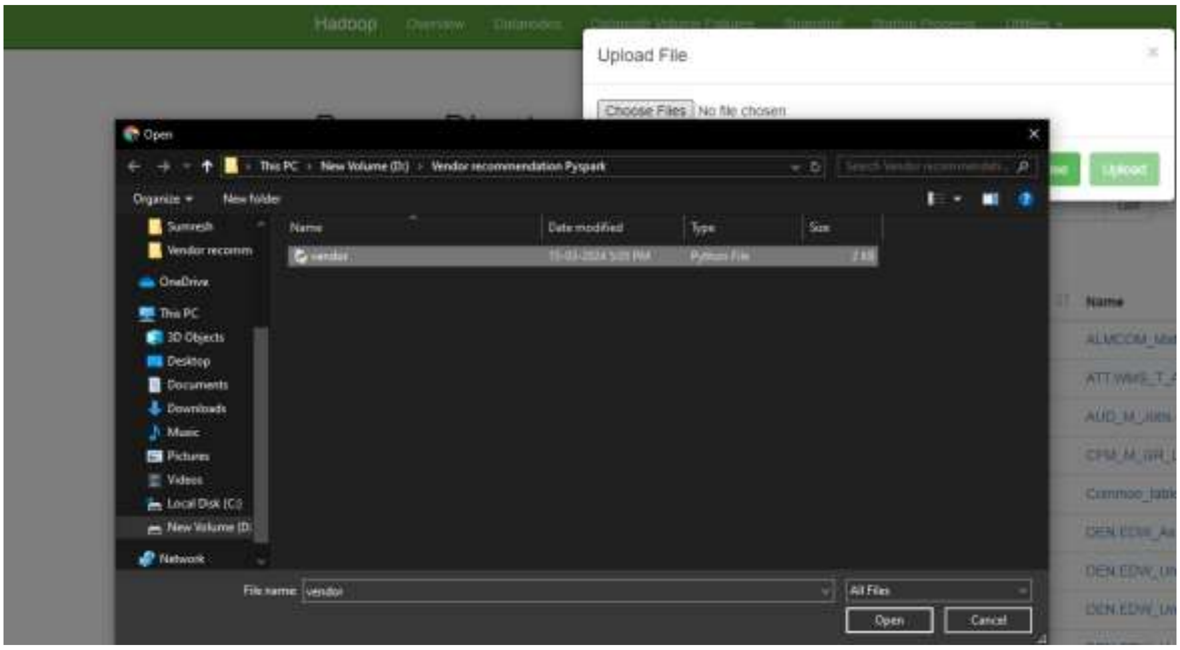
```
import streamlit as st

from Vendor_Recommendation_Steel_Data import main

st.title("Vendor Portal")

# Add a name or label for the textbox
input_value = st.text_input("Enter HPO_BA_CODE")
```

```
if not input_value:
    st.error("Please enter HPO_BA_CODE.")
else:
    # Enter DPO_Material_Code
    dpo = st.text_input("Enter DPO_Material_Code")

    if not dpo:
        st.error("Please enter DPO_Material_Code.")
    else:
        # Enter MMAT_Material_Description
        mmat = st.text_input("Enter MMAT_Material_Description")

        if not mmat:
            st.error("Please enter MMAT_Material_Description.")
        else:
            # Enter MMAT_Material_Category
            mmat2 = st.text_input("Enter MMAT_Material_Category")

            if not mmat2:
                st.error("Please enter MMAT_Material_Category.")
            else:
                # Enter MMC_DESCRIPTION
                mpem = st.text_input("Enter MMC_DESCRIPTION")

                if not mpem:
                    st.error("Please enter MMC_DESCRIPTION.")
                else:
                    # Enter MG_Country_Code
                    mg = st.text_input("Enter MG_Country_Code")

                    if not mg:
                        st.error("Please enter MG_Country_Code.")
```

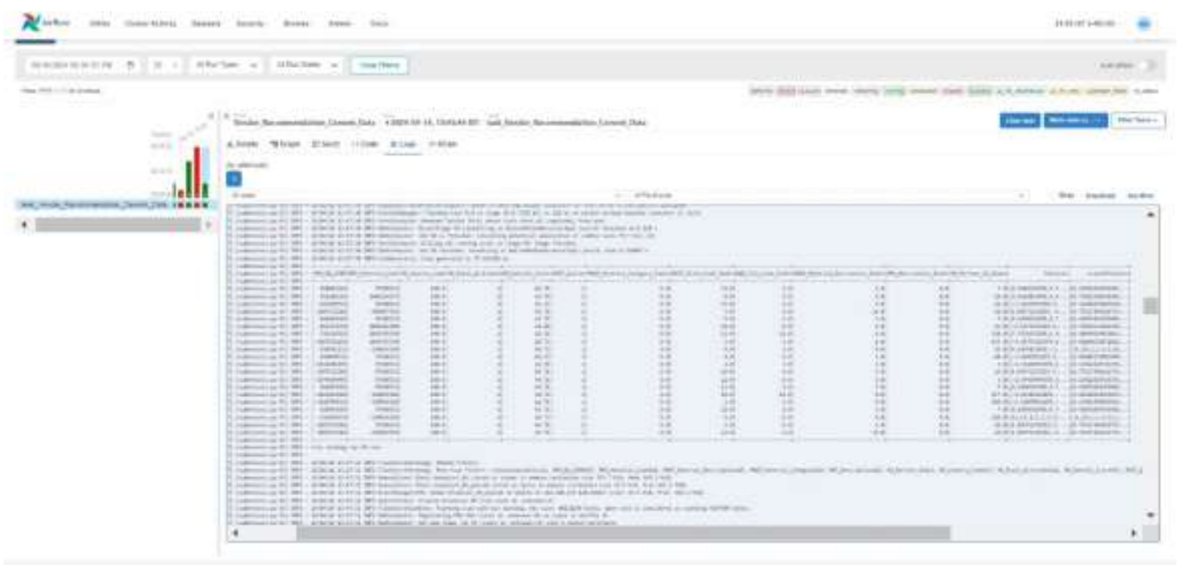
```
else:
    # Enter MG_EMAIL_ACTIVATED
    mab = st.text_input("Enter MG_EMAIL_ACTIVATED")

    if not mab:
        st.error("Please enter MG_EMAIL_ACTIVATED.")
    else:
        # Enter MG_OVERALL_SCORE
        mabs = st.text_input("Enter MG_OVERALL_SCORE")

        if not mabs:
            st.error("Please enter MG_OVERALL_SCORE.")
        else:
            # Enter MGST_ACTIVE
            mab2 = st.text_input("Enter MGST_ACTIVE")

            if not mab2:
                st.error("Please enter MGST_ACTIVE.")
            else:
                # Enter MGST_STATE_CODE
                mab3 = st.text_input("Enter MGST_STATE_CODE")

                if not mab3:
                    st.error("Please enter MGST_STATE_CODE.")
                else:
                    # Enter Enter MAB_CITY_CODE
                    maba = st.text_input("Enter MAB_CITY_CODE")

                    if not maba:
                        st.error("Please enter MAB_CITY_CODE.")

                    else:
```

```
        # Once all inputs are collected, create input_data dictionary
        input_data = {
            'HPO_BA_CODE': input_value,
            'DPO_Material_Code': dpo,
            'MMAT_Material_Description': mmat,
            'MMAT_Material_Category': mmat2,
            'MMC_DESCRIPTION': mpem,
            'MG_Country_Code': mg,
            'MG_EMAIL_ACTIVATED': mab,
            'MG_OVERALL_SCORE': mabs,
            'MGST_ACTIVE': mab2,
            'MGST_STATE_CODE': mab3,
            'Enter MAB_CITY_CODE': maba,
            }

            # Define a function to predict MG_Partner_Id
        resp =
requests.post('http://192.168.157.128:9871//api/experimental/dags/Vendor_Recommendati
on_Steel_Data/dag_runs')

            # Add a button to trigger the prediction
        if st.button('Predict MG_Partner_Id'):

                # Call the predict_mg_partner_id function
                mg_partner_id = predict_mg_partner_id(input_data)

                # Display the predicted MG_Partner_Id
                st.write(f"The predicted MG_Partner_Id is:
{mg_partner_id}")
```

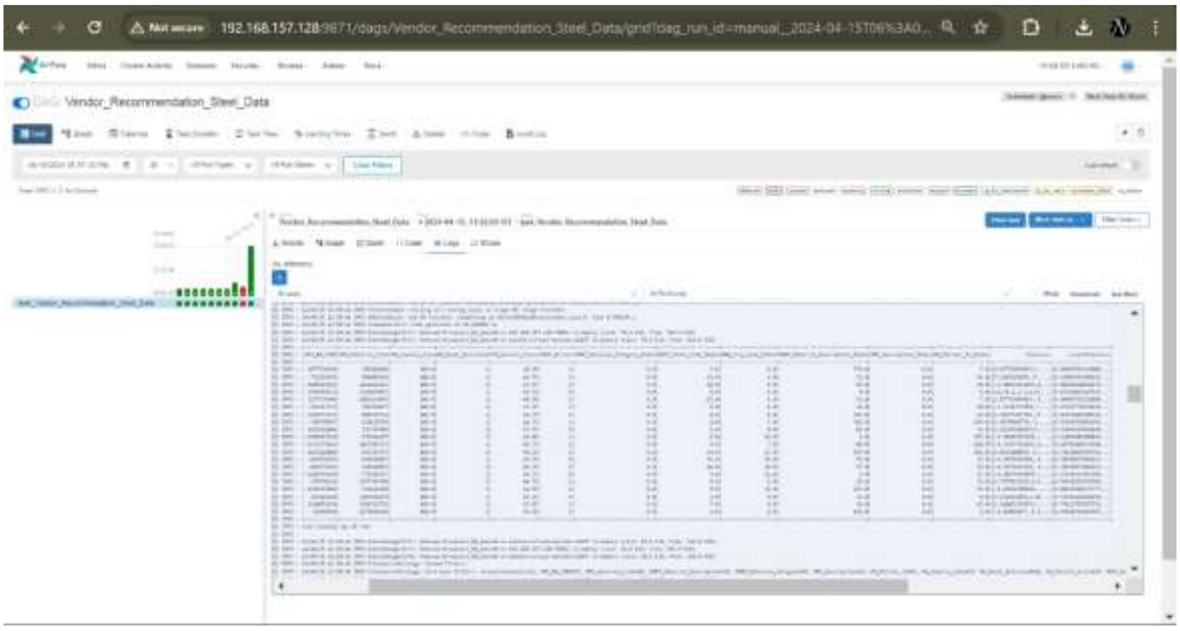
APPENDIX-2: SCREENSHOTS



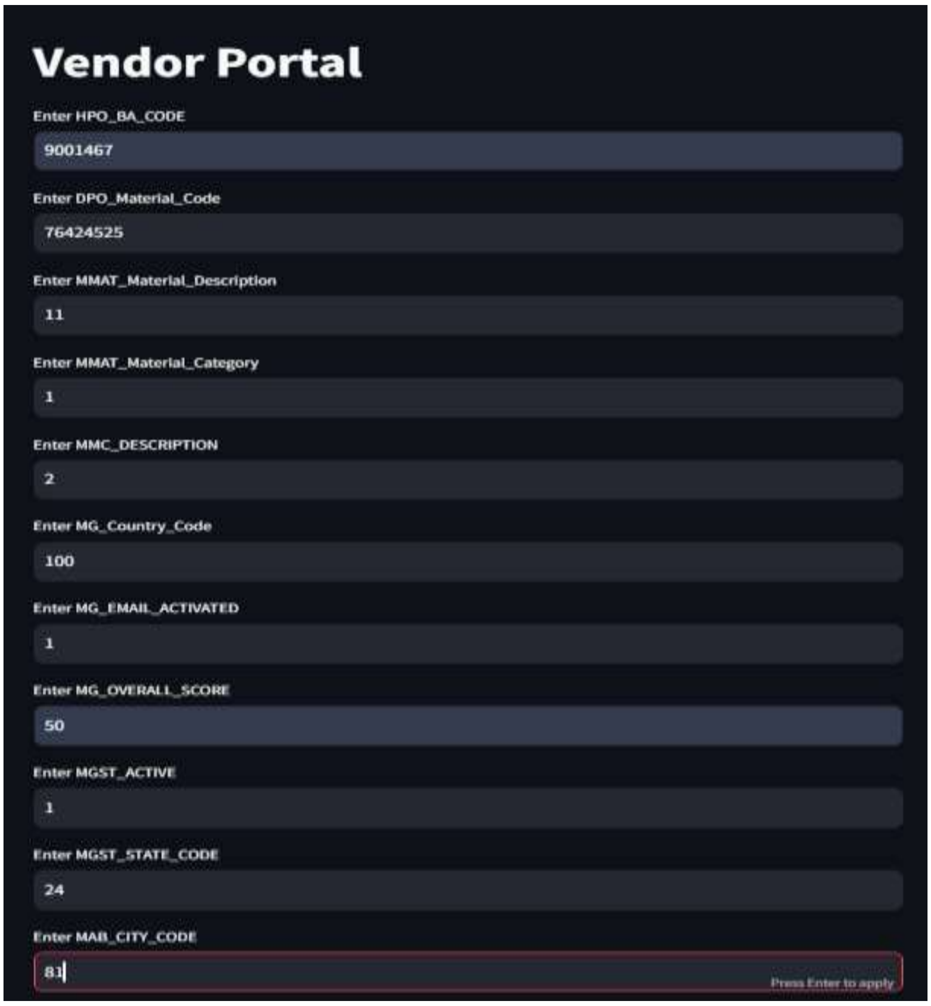
SS 1-- Hdfs



SS 2-Airflow (Cement Dag)


Not secure
192.168.157.128:8071/dags/Vendor_Recommendation_Steel_Data/grid?dag_run_id=manual__2024-04-15T06%3A0...
DAG Vendor_Recommendation_Steel_Data

SS 3-Airflow (Steel Dag



Vendor Portal

Enter HPO_BA_CODE	9001467
Enter DPO_Material_Code	76424525
Enter MMAT_Material_Description	11
Enter MMAT_Material_Category	1
Enter MMC_DESCRIPTION	2
Enter MG_Country_Code	100
Enter MG_EMAIL_ACTIVATED	1
Enter MG_OVERALL_SCORE	50
Enter MGST_ACTIVE	1
Enter MGST_STATE_CODE	24
Enter MAB_CITY_CODE	81

SS 4-Vendor Portal

REFERENCES

Journal References

1. Wang, C.H., 2015. Using quality function deployment to conduct vendor assessment and supplier recommendation for business-intelligence systems. Computers & Industrial Engineering, 84, pp.24-31.
2. Sabri, Y., Micheli, G.J. and Cagno, E., 2022. Supplier selection and supply chain configuration in the projects environment. Production planning & control, 33(12), pp.1155-1172.
3. Alrumiah, S.S. and Hadwan, M., 2021. Implementing big data analytics in e-commerce: Vendor and customer view. Ieee Access, 9, pp.37281-37286.
4. Peng, Y., 2022. A survey on modern recommendation system based on big data. arXiv preprint arXiv:2206.02631.
5. Cheng, J. and Zhao, P., 2023. Sustainable Big Data Analytics Process Pipeline Using Apache Ecosystem. In Encyclopedia of Data Science and Machine Learning (pp. 1247-1259). IGI Global.

Web References

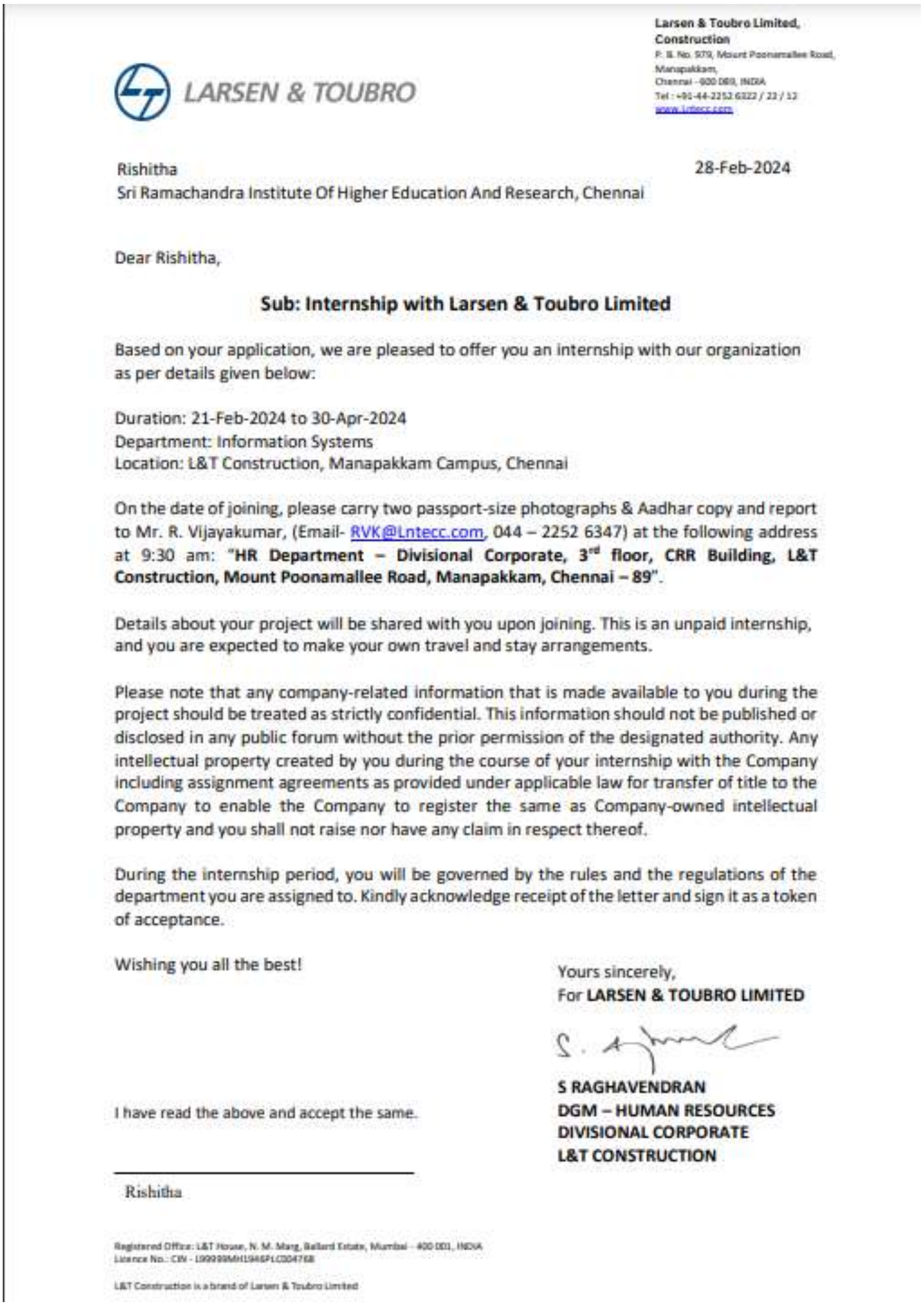
1. PySpark Tutorial – javatpoint
2. https://www.youtube.com/watch?v=5peQThvQmQk
3. https://www.youtube.com/watch?v=6kEGUCrBEU0&t=257s

WORKLOG

Day	Date	Task Done
Day 1	29/02/2024	Day of Joining (Finishing Formalities), Introduction to Pyspasrk
Day 2	01/03/2024	Discussion of Vendor Recommendation project
Day 3	04/03/2024	Installation of PySpark
Day 4	05/03/2024	Task 1 and Task 2 done (Given by Company)
Day 5	06/03/2024	Learning Pyspark Data Preparation and executing Project
Day 6	07/03/2024	Learning Pyspark Data Preprocessing and executing Project
Day 7	08/03/2024	Data Preprocessing
Day 8	11/03/2024	Attended Webinar
Day 9	12/03/2024	Data Normalisation
Day 10	13/03/2024	Airflow Presentation (In Company)
Day 11	14/03/2024	Learning about Flask Api
Day 12	15/03/2024	Installation of Hadoop
Day 13	16/03/2024	First Review
Day 14	18/03/2024	Feature scaling data
Day 15	19/03/2024	Coding the project
Day 16	20/03/2024	Correcting errors in data normalization & Data Standardization
Day 17	21/03/2024	Learning about ml algorithms in detail
Day 18	22/03/2024	Applying ml algorithm to the data frame (logistic reg)
Day 19	25/03/2024	Researching about pyspark data analysis projects

Day	Date	Task Done
Day 20	26/03/2024	Correcting errors in ml algorithm code
Day 21	27/03/2024	Coding the ml algorithms
Day 22	01/04/2024	Learning about ANN
Day 23	02/04/2024	Completed task given by company
Day 24	03/04/2024	Coding ML Algorithms
Day 25	04/04/2024	Completed tasks given by company
Day 26	05/04/2024	Second Review
Day 27	08/04/2024	Learning about Fast Api
Day 28	09/04/2024	Learning about Streamlit
Day 29	11/04/2024	Selection of the Algorithm code
Day 30	12/04/2024	Creating webpage using Streamlit
Day 31	15/04/2024	Integrating Steel and Cement files with webpage
Day 32	16/04/2024	Preparing Report
Day 33	17/04/2021	Completion of project
Day 34	18/04/2021	Final Review

OFFER LETTER



LARSEN & TOUBRO

Larsen & Toubro Limited,
Construction
P. B. No. 979, Mount Poonamallee Road,
Manapakkam,
Chennai - 600 089, INDIA
Tel : +91-44-2252 6322 / 23 / 12
www.Lntecc.com

Rishitha
Sri Ramachandra Institute Of Higher Education And Research, Chennai

28-Feb-2024

Dear Rishitha,

Sub: Internship with Larsen & Toubro Limited

Based on your application, we are pleased to offer you an internship with our organization as per details given below:

Duration: 21-Feb-2024 to 30-Apr-2024
Department: Information Systems
Location: L&T Construction, Manapakkam Campus, Chennai

On the date of joining, please carry two passport-size photographs & Aadhar copy and report to Mr. R. Vijayakumar, (Email- RVK@Lntecc.com, 044 – 2252 6347) at the following address at 9:30 am: **"HR Department – Divisional Corporate, 3rd floor, CRR Building, L&T Construction, Mount Poonamallee Road, Manapakkam, Chennai – 89"**.

Details about your project will be shared with you upon joining. This is an unpaid internship, and you are expected to make your own travel and stay arrangements.

Please note that any company-related information that is made available to you during the project should be treated as strictly confidential. This information should not be published or disclosed in any public forum without the prior permission of the designated authority. Any intellectual property created by you during the course of your internship with the Company including assignment agreements as provided under applicable law for transfer of title to the Company to enable the Company to register the same as Company-owned intellectual property and you shall not raise nor have any claim in respect thereof.

During the internship period, you will be governed by the rules and the regulations of the department you are assigned to. Kindly acknowledge receipt of the letter and sign it as a token of acceptance.

Wishing you all the best!

Yours sincerely,
For **LARSEN & TOUBRO LIMITED**

[illegible]

S RAGHAVENDRAN
DGM – HUMAN RESOURCES
DIVISIONAL CORPORATE
L&T CONSTRUCTION

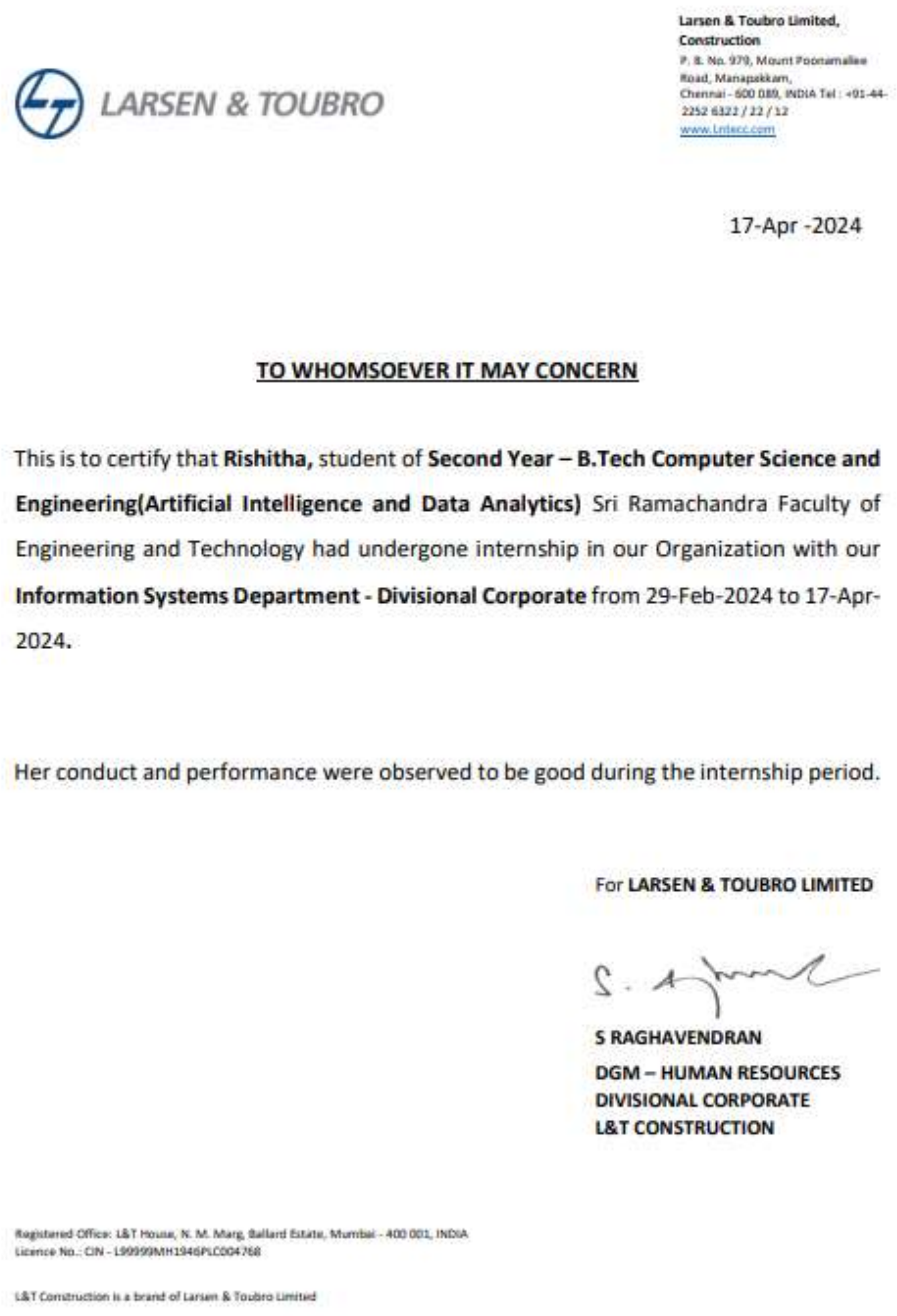
I have read the above and accept the same.

Rishitha

Registered Office: L&T House, N. M. Marg, Ballard Estate, Mumbai - 400 001, INDIA
Licence No.: CIN - L99999MH1946PLC004768

L&T Construction is a brand of Larsen & Toubro Limited

CERTIFICATE OF COMPLETION



LARSEN & TOUBRO

Larsen & Toubro Limited,
Construction
P. B. No. 979, Mount Poonamallee
Road, Manapakkam,
Chennai - 600 089, INDIA Tel : +91-44-
2252 6321 / 22 / 12
www.Lntecc.com

17-Apr -2024

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Rishitha,** student of **Second Year – B.Tech Computer Science and Engineering(Artificial Intelligence and Data Analytics)** Sri Ramachandra Faculty of Engineering and Technology had undergone internship in our Organization with our **Information Systems Department - Divisional Corporate** from 29-Feb-2024 to 17-Apr-2024.

Her conduct and performance were observed to be good during the internship period.

For **LARSEN & TOUBRO LIMITED**

[illegible]

S RAGHAVENDRAN
DGM – HUMAN RESOURCES
DIVISIONAL CORPORATE
L&T CONSTRUCTION

Registered Office: L&T House, N. M. Marg, Ballard Estate, Mumbai - 400 001, INDIA
Licence No.: CIN - L99999MH1946PLC004768

L&T Construction is a brand of Larsen & Toubro Limited