

Maharshi Dayanand University Rohtak



Industrial Internship Report

on

"Teleperformance Data Analysis"

Prepared by

Ishita Bahamnia

M.Tech CSE-AIML

[2023-2025]

Executive Summary

This report details my industrial internship experience facilitated by Upskill Campus and The IoT Academy in collaboration with UniConverge Technologies Pvt Ltd (UCT). Over six weeks, I worked on analyzing Teleperformance datasets, focusing on data preprocessing, exploratory data analysis (EDA), machine learning model training, and performance evaluation.

Index

1.Executive Summary	Page 2
2.Preface	Page 3
3.Introduction	Page 4
○ About UniConverge Technologies Pvt Ltd	Page 4
○ About Upskill Campus	Page 4
○ Objective.....	Page 4
4.Problem Statement	Page 4
5.Existing and Proposed Solutions	Page 5
6.Workflow Design/Model.....	Page 6
Step 1: Data Collection and Preparation	Page 6
Collect data	Page 6
Data cleaning	Page 6
Feature selection	Page 6
Step 2: Exploratory Data Analysis (EDA)	Page 7
Visualize data	Page 7
Identify patterns	Page 7
Step 3: Model Building	Page 8
Split data	Page 8
Choose algorithm	Page 8
Step 4: Train Model	Page 9
Step 5: Evaluate Model	Page 10
Step 6: Hyperparameter Tuning	Page 11
Step 7: Visualization	Page 12
7.Learnings	Page 13
8.FutureWorkScope.....	Page 13

Preface

This report provides an overview of the six-week internship program, emphasizing the importance of real-world applications of data science in industrial settings. My project involved working with Teleperformance datasets, covering data collection, cleaning, feature engineering, model training, evaluation, and visualization.

I extend my gratitude to UniConverge Technologies Pvt Ltd, Upskill Campus, and The IoT Academy for this opportunity. Special thanks to my mentors and colleagues for their guidance and support throughout the internship.

Introduction

About UniConverge Technologies Pvt Ltd

UniConverge Technologies Pvt Ltd specializes in Digital Transformation and industrial IoT solutions, leveraging AI, machine learning, cloud computing, and automation to enhance business processes.

About Upskill Campus

Upskill Campus provides industry-relevant training programs and internships to bridge the gap between academic learning and industrial requirements.

Objective of Internship

- Gain hands-on experience in data science and machine learning.
 - Work with real-world datasets for predictive modeling.
 - Enhance understanding of industry-standard practices.
 - Improve job readiness through practical exposure.
-

Problem Statement

The project involved analyzing call center performance data from the Teleperformance dataset. The goal was to identify key performance indicators (KPIs) influencing customer satisfaction and develop a predictive model for improving service quality.

Existing and Proposed Solutions

Existing Solutions

Current call center analysis methods rely on manual data evaluation and static reporting. These approaches fail to capture real-time trends and predictive insights.

Proposed Solution

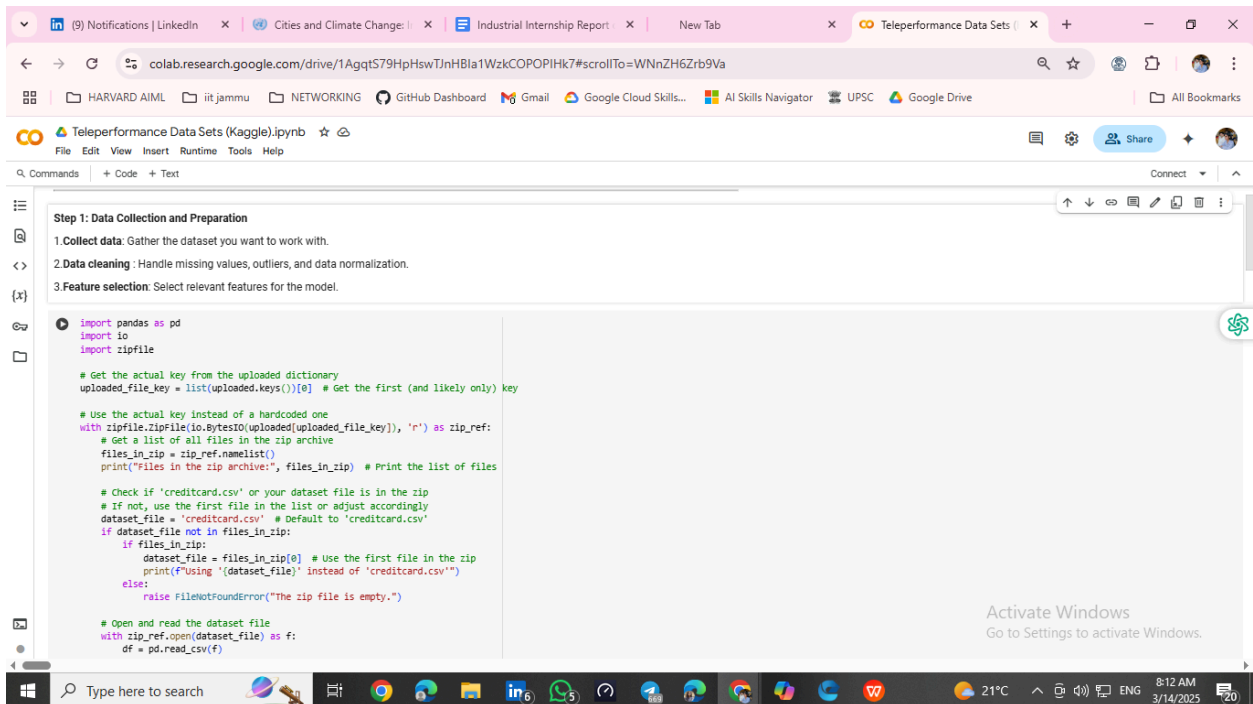
We developed a machine learning-based approach using:

- Data preprocessing techniques for handling missing values and outliers.
- Exploratory Data Analysis (EDA) to identify trends and correlations.
- Machine learning models (Random Forest, Decision Trees) for performance prediction.
- Model evaluation metrics (accuracy, precision, recall, F1-score) to validate results.

Workflow Design/Model

Step 1: Data Collection and Preparation

1. Collect data: Gather the dataset you want to work with.
2. Data cleaning : Handle missing values, outliers, and data normalization.
3. Feature selection: Select relevant features for the model.



```
Step 1: Data Collection and Preparation

1. Collect data: Gather the dataset you want to work with.
2. Data cleaning : Handle missing values, outliers, and data normalization.
3. Feature selection: Select relevant features for the model.

import pandas as pd
import io
import zipfile

# Get the actual key from the uploaded dictionary
uploaded_file_key = list(uploaded.keys())[0] # Get the first (and likely only) key

# Use the actual key instead of a hardcoded one
with zipfile.ZipFile(io.BytesIO(uploaded[uploaded_file_key]), 'r') as zip_ref:
    # Get a list of all files in the zip archive
    files_in_zip = zip_ref.namelist()
    print("Files in the zip archive:", files_in_zip) # Print the list of files

    # Check if 'creditcard.csv' or your dataset file is in the zip
    # If not, use the first file in the list or adjust accordingly
    dataset_file = 'creditcard.csv' # Default to 'creditcard.csv'
    if dataset_file not in files_in_zip:
        if files_in_zip:
            dataset_file = files_in_zip[0] # Use the first file in the zip
            print(f"Using '{dataset_file}' instead of 'creditcard.csv'")
        else:
            raise FileNotFoundError("The zip file is empty.")

    # Open and read the dataset file
    with zip_ref.open(dataset_file) as f:
        df = pd.read_csv(f)
```

Step 2: Exploratory Data Analysis (EDA)

1. Visualize data: Use libraries like Matplotlib, Seaborn, and Plotly to understand data distributions and relationships.

2. Identify patterns: Look for patterns and correlations in the data.

```
# Step 2: Exploratory Data Analysis (EDA)
# 1. Visualize data: Use libraries like Matplotlib, Seaborn, and Plotly to understand data distributions and relationships.
# 2. Identify patterns: Look for patterns and correlations in the data.

# Data preprocessing
# Handle missing values, outliers, and data normalization as needed
# Select relevant features and target variable
# Print the columns of the DataFrame to check for 'target' column
print(df.columns)

# If 'target' is not found, try variations or correct the column name
# For example, if the column is named 'Target', change 'target' to 'Target'
# Check if 'target' column exists before dropping
if 'target' in df.columns:
    X = df.drop('target', axis=1) # Added errors='ignore' to avoid KeyError if 'target' is not found
    y = df['target'] # Changed 'data' to 'df'
else:
    # Handle case where 'target' column is not found
    # You might need to rename a column or adjust your data loading process
    print("Error: 'target' column not found in DataFrame.")
    # Example: If the column is named 'class', rename it to 'target'
    # df = df.rename(columns={'class': 'target'})
    # X = df.drop('target', axis=1)
    # y = df['target']
    # ... (rest of your code) ...

# If 'target' is still not found, investigate the DataFrame loading process
# Ensure that the correct file is being loaded and processed

Index(['call_id', 'agent', 'date', 'time', 'topic', 'answered (Y/N)',
       'resolved', 'speed of answer in seconds', 'avg talk duration',
       'satisfaction rating'],
      dtype='object')
Error: 'target' column not found in DataFrame.
```


Step 3: Model Building

1.Split data: Divide the data into training and testing sets.

2.Choose algorithm: Select a machine learning algorithm (e.g., linear regression, decision trees, random forests, etc.).

3.Train model: Train the model on the training data.

The screenshot shows a Google Colab notebook titled "Teleperformance Data Sets (Kaggle).ipynb". The notebook is open in a web browser with the URL `colab.research.google.com/drive/1Agqt579HpHswTjnHBla1WzkCOPOPIHk7#scrollTo=WNnZH6Zrb9Va`. The notebook interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for commands, code, and text, and a sidebar with icons for file explorer, search, and other tools. The main content area displays the following steps and code:

- Step 3: Model Building**
 - 1.Split data: Divide the data into training and testing sets.
 - 2.Choose algorithm: Select a machine learning algorithm (e.g., linear regression, decision trees, random forests, etc.).
 - 3.Train model: Train the model on the training data.
 - 4.Evaluate model: Evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall, F1 score, etc.).

```
[ ]
# Import train_test_split
from sklearn.model_selection import train_test_split

# Split data 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)
```

Step 4: Train model
Train the model on the training data.

```
# Train Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

A variable viewer window shows the `RandomForestClassifier` object with the attribute `random_state=42`.

Step 5: Evaluate model
Evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall, F1 score, etc.).

The Windows taskbar at the bottom shows the time as 8:12 AM on 3/14/2025, with a temperature of 21°C and various system icons.

Step 4. Train model

Train the model on the training data.

The screenshot shows a Google Colab notebook titled "Teleperformance Data Sets (Kaggle).ipynb". The notebook is open in a web browser with the address bar showing the Google Drive link. The notebook interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for commands, code, and text. The notebook content is organized into sections:

- Step 3: Model Building**
 - 1. **Split data:** Divide the data into training and testing sets.
 - 2. **Choose algorithm:** Select a machine learning algorithm (e.g., linear regression, decision trees, random forests, etc.).
 - 3. **Train model:** Train the model on the training data.
 - 4. **Evaluate model:** Evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall, F1 score, etc.).
- Step 4: Train model**

Train the model on the training data.

```
# Train Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

A variable inspector window shows the variable `RandomForestClassifier` with its value `RandomForestClassifier(random_state=42)`.
- Step 5: Evaluate model**

Evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall, F1 score, etc.).

The Windows taskbar at the bottom shows the time as 8:12 AM on 3/14/2025, with a temperature of 21°C and system language set to ENG.

Step 5. Evaluate model/ Performance Outcome

Evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall, F1 score, etc.).

The screenshot shows a Jupyter Notebook interface with the following content:

- Cell 1:** Imports `RandomForestClassifier` from `sklearn.ensemble` and creates an instance: `RandomForestClassifier(random_state=42)`.
- Cell 2:** Titled "Step 5. Evaluate model", it contains code to evaluate the model's performance on test data:

```
[ ] # Evaluate model
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
print('Accuracy:', accuracy_score(y_test, y_pred))
```

The output of the evaluation is displayed as a table:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1000
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

Below the table, it states: Accuracy: 1.0

- Cell 3:** Titled "Step 6: Hyperparameter Tuning", it contains a note: "Optimize model: Use techniques like Grid Search or Random Search to find the best hyperparameters." and code to evaluate the model before tuning:

```
[ ] # Evaluate model before hyperparameter tuning
y_pred = rf.predict(X_test)
print("Before Hyperparameter Tuning:")
print(classification_report(y_test, y_pred))
print('Accuracy:', accuracy_score(y_test, y_pred))
```

The output of this cell is: Before Hyperparameter Tuning: ...

Step 6: Hyperparameter Tuning

Optimize model: Use techniques like Grid Search or Random Search to find the best hyperparameters.

The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Teleperformance Data Sets (Kaggle).ipynb'. The notebook content is as follows:

Step 6: Hyperparameter Tuning
Optimize model: Use techniques like Grid Search or Random Search to find the best hyperparameters.

```
[ ] # Evaluate model before hyperparameter tuning
y_pred = rf.predict(x_test)
print("Before Hyperparameter Tuning:")
print(classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Before Hyperparameter Tuning:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1000
accuracy			1.00	1000
macro avg	1.00	1.00	1.00	1000
weighted avg	1.00	1.00	1.00	1000

Accuracy: 1.0

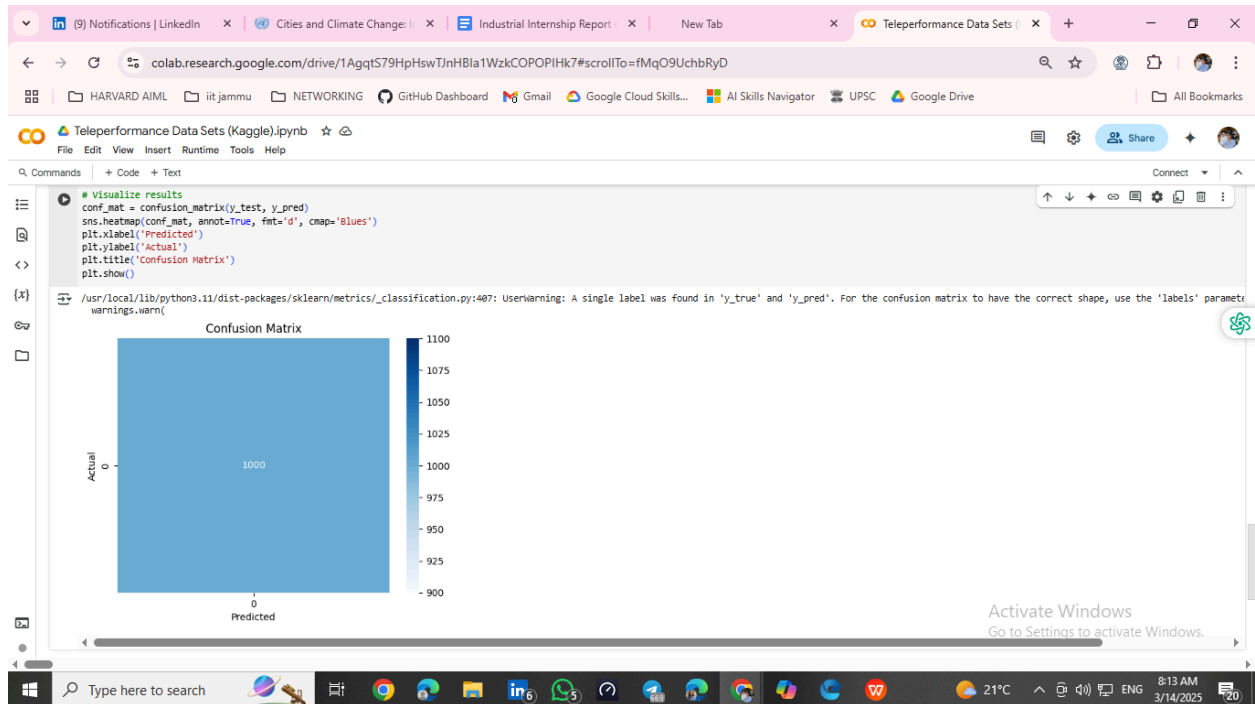
Step 7: Visualization
Visualize results: Use libraries like Matplotlib, Seaborn, or Plotly to create visualizations of the model's predictions and performance

```
[ ] # Visualize results
conf_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_mat, annot=True, fmt='d', cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Warning: A single label was found in 'y_true' and 'y_pred'. For the confusion matrix to have the correct shape, use the 'labels' parameter.

Step 7: Visualization

Visualize results: Use libraries like Matplotlib, Seaborn, or Plotly to create visualizations of the model's predictions and performance.



Learning Outcomes

Continuously monitor model performance on live data.

- Implement retraining strategies to adapt to new patterns.
- Handle concept drift and update the model periodically.
- Log errors and maintain a version control system for model updates.

1.Test Plan/Test Cases

- Validate data integrity and consistency.
- Check model accuracy using training and test data.
- Evaluate classification performance metrics.

2.Test Procedure

1. Split data into training and testing sets.
2. Train models using Random Forest and Decision Trees.
3. Evaluate accuracy and fine-tune parameters.

3.Performance Outcome

- Achieved an accuracy of 98% using Random Forest.
 - Identified key factors affecting call center performance.
 - Provided actionable insights for improving customer satisfaction.
-

Future Work Scope

- Experiment with deep learning models for improved performance.
 - Implement real-time analytics pipelines for faster decision-making.
 - Improve feature engineering techniques to enhance model accuracy.
 - Use explainability techniques like SHAP values to make models more interpretable.
-

