A

Assesment Report on

"Predict Employee Attrition"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25 in

CSEAIML

By

Divyanshi Verma(20240110040088)

Under the supervision of

"Mr. Abhishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU) **May, 2025**

Introduction

This project focuses on developing a machine learning model to predict **customer churn** in a telecom company based on customer usage patterns and demographics. Churn prediction helps telecom companies identify customers who are likely to leave and implement strategies to retain them.

2. Objectives

- **To preprocess and clean real-world customer churn data.**
- **To train a reliable and interpretable machine learning model.**
- **To evaluate the model using appropriate classification metrics.**
- **To save the trained model for future deployment and reuse.**

4. Data Preprocessing

- **Missing values handled by filling numerical columns with median and categorical columns with mode.**
- **Encoding performed using one-hot encoding for categorical variables.**
- **Feature Scaling using StandardScaler from Scikit-learn.**

Code

```
# Step 1: Upload the dataset

from google.colab import files
```

```
import pandas as pd
import io

# Upload CSV file
uploaded = files.upload()
file_name = next(iter(uploaded))
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

# Add 'Class' column if not present
if 'Class' not in df.columns:
    # Try to use 'Churn' as the target if it exists
    churn_col = next((col for col in df.columns if col.lower().strip() == 'churn'), None)
    if churn_col:
        df['Class'] = df[churn_col].map({'No': 0, 'Yes': 1})
    else:
        raise ValueError("The dataset must contain a 'Class' column or a 'Churn' column to be used as the target.")
```

```
# Step 2: Validate dataset
if 'Class' not in df.columns:
    raise ValueError("The dataset must contain a 'Class' column as the target variable.")

# Step 3: Preprocess the dataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    confusion_matrix,
    classification_report,
    accuracy_score,
    precision_score,
    recall_score
)
import seaborn as sns
```

```
import matplotlib.pyplot as plt
import joblib


# Drop non-numeric and non-useful columns (e.g., IDs)
df = df.select_dtypes(include=['int64', 'float64',
'bool']).copy()


# Feature selection and scaling
X = df.drop('Class', axis=1)
y = df['Class']


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42
)
```

```
# Train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, zero_division=0)
recall = recall_score(y_test, y_pred, zero_division=0)

print(f"\n☑ Accuracy: {accuracy:.4f}")
print(f"☑ Precision: {precision:.4f}")
print(f"☑ Recall: {recall:.4f}")
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Churn', 'Churn'],
            yticklabels=['Not Churn', 'Churn'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('🔍 Confusion Matrix - Heatmap')
plt.show()

# Detailed classification report
print("\n📊 Classification Report:\n")
print(classification_report(y_test, y_pred,
zero_division=0))

# Save model and scaler
```

```
joblib.dump(model, 'churn_classifier_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

print("\n💾 Model and scaler saved successfully.")
```

References/Credits

- Dataset: IBM HR Analytics Employee Attrition & Performance Dataset from Kaggle
- Libraries: Scikit-learn, Imbalanced-learn, Pandas, NumPy
- Special thanks to course instructors and online documentation