Srinivas Bhootam

DeVos Graduate School, Northwood University

115537-MGT-665-NW Solv Probs W/ Machine Learning

Lab 2: Predicting Customer Churn: A Comparative Study of Logistic Regression, k-NN, and Decision Tree Models

Abstract :

In this research supervised machine learning algorithms has been utilized for predicting the Customer Churn for the Telco Customer Churn dataset. Churn, when customers desert one service, is a key problem for telecom companies. I will Construct and compare performance metrics of following 3 classification models they are Logistic Regression, k-NN and Decision Tree. I will preprocess, visualize, and analyze the dataset before training and evaluating models. In the three models, the Logistic Regression is the model with the most balanced accuracy across the measures. The paper provides a discussion on the steps of model development, insights into evaluation, and the outlook for which improvements could be introduced. Such in-sights can be used by telecommunication companies to report and forecast churn to minimize its impact on profitability.

Introduction :

Customer churn prediction is particularly useful for companies that provide recurring services, such as telecom companies, since retaining customers is often less expensive than acquiring new ones. Once a company knows which of its customers are most likely to leave, it can take steps to keep those customers by offering them better deals or posture with better service. In this paper I will develop and compare three machine learning models – Logistic Regression, k-Nearest Neighbors (k-NN), and Decision Tree, using actual customer data set from a telecom company. These models allow us to discern patterns in the data that indicate what customers are most likely to churn. Also compare how well each model does using basic metrics like accuracy, precision, recall, and F1-score and then decide which model is ideal for predicting churn and assisting companies to minimize customer attrition.

Literature review :

Customer churn has been well researched for years; it's a significant problem for firms who depend on repeat business. Indeed, some researchers such as Hadden et al. (2007) demonstrated that computerized decision making can improve the ability of companies to learn about why customers take their business elsewhere and respond appropriately. Logistic Regression is a common model used for this purpose since it is easy to use and reveals what increases the odds of a customer leaving (Han, Kamber, & Pei, 2011).

Other models such as k-Nearest Neighbors (k-NN) and Decision Trees are also utilized. k-NN is easy to use but does not perform well if the data is unbalanced or not properly scaled (Ahmed et al., 2016). Decision Trees can manage a wide range of different types of data, providing simple and easy-to-understand decision rules, while at the same time they can overfit and not generalize properly for new data (Pedregosa et al., 2011). Though more recent approaches such as deep learning make little sense economically, these simpler models remain valuable because they are fast, easy to understand, and effective in numerous commercial applications.

Methodology :

The Telco Customer Churn dataset consists of the following 7,043 rows and 21 columns, which contain information about multiple aspects of the customer such as their demographic information, the services that they have subscribed to, contract type, and monthly charges: The response variable, Churn, is a customer becoming not being a customer.

Pre-analysis Data processing: CustomerID column was removed as it doesn't contribute towards prediction. The TotalCharges column had empty strings or strings entries, which were changed to numeric and any NA's deleted. Categorical values were converted to numerical values using labels encoding them to make them applicable to machine learning.

Once the data was cleaned and encoded, it was split into input features (X) and the target variable (y). 80% and 20% division ratio was used for splitting the data into training and testing sets. As certain algorithms (e.g. Logistic Regression, k-Nearest Neighbors) are sensitive to feature scaling, we scaled the numeric columns with StandardScaler. This step was taken to make sure that variables such as tenure and MonthlyCharges contributed equally to the learning.

Models Description :

In this paper, three supervised machines learning algorithms such as Logistic Regression, k-Nearest Neighbors (k-NN), and Decision Tree Classifier were used to predict customer churn. We chose these two models, because they are universally used for classification tasks, simple to implement, and interpretable.

Logistic Regression (LR). Also known as Logit Regression, LR is a statistical method for analyzing a data set in which there are one or more independent variables that determine an outcome: in this example if a customer will or will not churn. It models the relationship between the dependent and one or more independent variable using a logistic (sigmoid) function. This model is especially convenient to interpret when it comes to the importance and direction of impact of each characteristic on the probability of the churn.

k-Nearest Neighbors (k-NN) is a simple non-parametric, instance-based learning algorithm that labels a new observation according to the majority class of its nearest neighbors. In this study, the k value was 5, so the prediction was calculated using the five most similar data in the training dataset. Simple though it is, the model performs subject to feature rescaling and data imbalance.

Decision Tree Classifier is another model based on decision trees that cuts recursively a set of data in subsets with a decision rule taking each subset. Every internal node is a decision on a feature, while each leaf node is a class label prediction. Even though the decision trees are highly interpretable, with low level models they have the same issue of overfitting on the (small) dataset, unless you have set high value for appropriate parameters.

Analysis :

Step 1: Import Necessary Libraries

This step involves importing all the essential libraries required for the churn prediction project. These include:

- `pandas` and `numpy` for data manipulation and numerical operations
- `matplotlib` and `seaborn` for data visualization
- `scikit-learn` modules for:
  - Splitting data into training and testing sets
  - Preprocessing tasks like encoding and scaling
  - Building classification models: Logistic Regression, k-NN, and Decision Tree
  - Evaluating model performance using classification metrics

In [4]:
```
# Step 1: Import necessary libraries
# These libraries are used for data manipulation, visualization, model buildin
g, and evaluation.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, Confusion
MatrixDisplay, f1_score
```

Step 2: Load the Dataset

In this step, the Telco Customer Churn dataset is loaded into a Pandas DataFrame. This dataset contains customer information such as demographic details, account information, and service usage patterns. It will be used to train and evaluate churn prediction models.

The `head()` function is used to display the first few rows of the dataset for a quick overview.

In [5]:
```
# Step 2: Load the dataset
# We're using the Telco Customer Churn dataset.
df = pd.read_csv("Telco-Customer-Churn.csv")
df.head()
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

5 rows × 21 columns

Step 3: Exploratory Data Analysis (EDA)

This step focuses on understanding the structure, patterns, and key relationships in the dataset. Visualizations are used to gain insights that may help improve model performance later.

- **Figure 1: Churn Distribution**
  A countplot is used to show how many customers have churned (1) vs. how many have not churned (0). This helps assess class imbalance in the dataset.
- **Figure 2: Histograms of Numerical Features**
  Two histograms display the distribution of tenure (in months) and monthly charges. This helps understand the spread and central tendency of key numerical features.
- **Figure 3: Monthly Charges vs. Churn**
  A boxplot visualizes how monthly charges vary between churned and non-churned customers. This may reveal if higher charges are associated with churn.

In [24]:
```
# Step 3: Exploratory Data Analysis (EDA)
# Let's understand the structure, distribution, and relationships in the data.

import matplotlib.pyplot as plt
import seaborn as sns

# 3.1 Churn Distribution (Figure 1)
fig, ax = plt.subplots()
sns.countplot(x='Churn', data=df, ax=ax)
ax.set_title("Churn Distribution")
ax.set_xlabel("Churn (0 = No, 1 = Yes)")
ax.set_ylabel("Count")
fig.subplots_adjust(bottom=0.2)
fig.text(0.5, 0.05,
         "Figure 1. Churn Distribution. Bar chart showing counts of non-churne
d vs churned customers.",
         ha='center', fontsize=10)
plt.show()

# 3.2 Histograms of Numerical Features (Figure 2)
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
axes[0].hist(df['tenure'], bins=20)
axes[0].set_title('Tenure Distribution')
axes[0].set_xlabel('Tenure (months)')
axes[0].set_ylabel('Frequency')

axes[1].hist(df['MonthlyCharges'], bins=20)
axes[1].set_title('Monthly Charges Distribution')
axes[1].set_xlabel('Monthly Charges')
axes[1].set_ylabel('Frequency')

fig.suptitle("Histograms of Numerical Features")
fig.subplots_adjust(bottom=0.2, top=0.85)
fig.text(0.5, 0.01,
         "Figure 2. Histograms showing distributions of customer tenure and mo
nthly charges.",
         ha='center', fontsize=10)
plt.tight_layout()
plt.show()

# 3.3 Monthly Charges vs Churn (Figure 3)
fig, ax = plt.subplots()
sns.boxplot(x='Churn', y='MonthlyCharges', data=df, ax=ax)
ax.set_title("Monthly Charges vs Churn")
ax.set_xlabel("Churn (0 = No, 1 = Yes)")
ax.set_ylabel("Monthly Charges")
fig.subplots_adjust(bottom=0.2)
fig.text(0.5, 0.05,
         "Figure 3. Boxplot of monthly charges by churn status.",
         ha='center', fontsize=10)
plt.show()
```

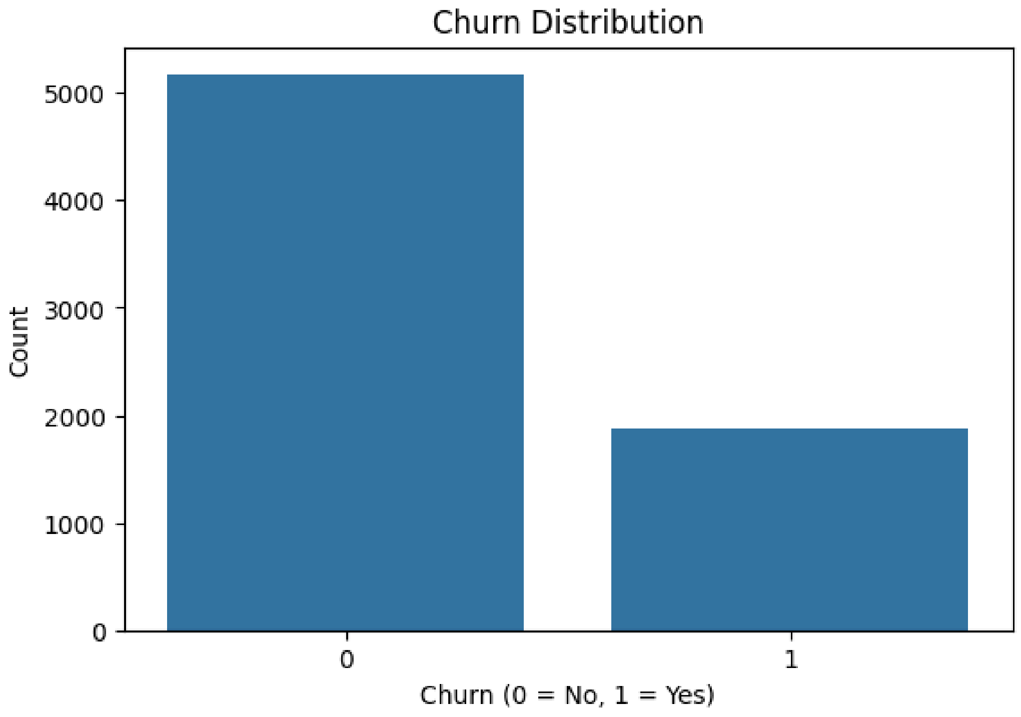

Figure 1. Churn Distribution. Bar chart showing counts of non-churned vs churned customers.

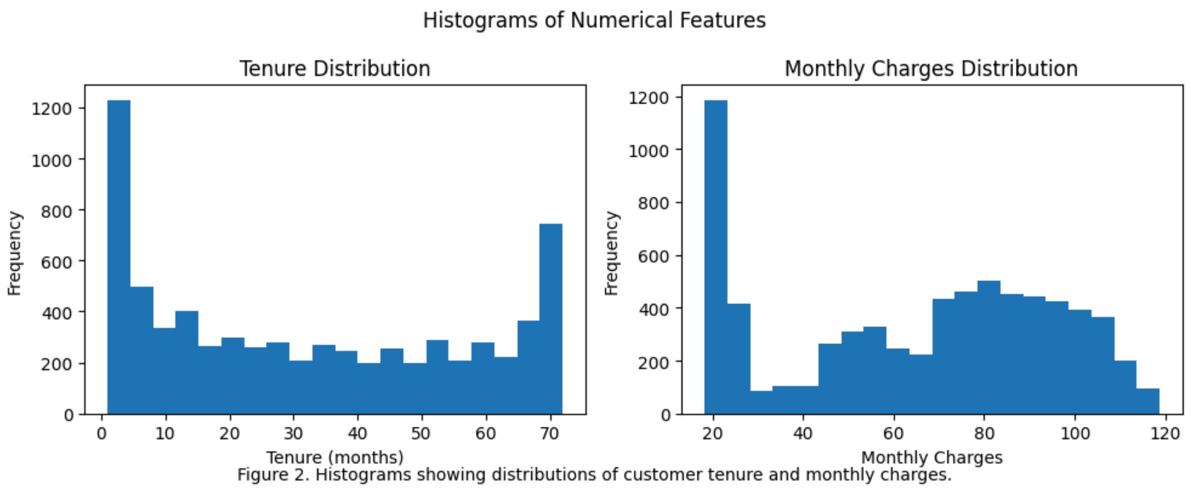

Figure 2. Histograms showing distributions of customer tenure and monthly charges.

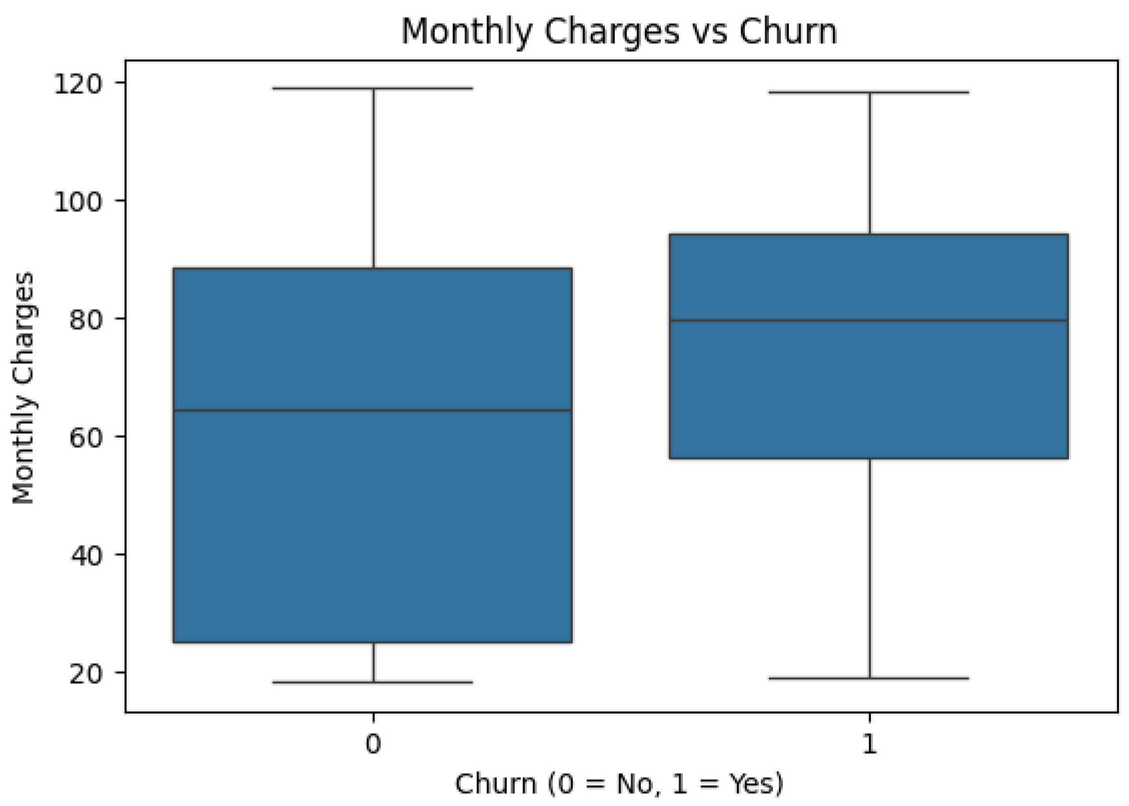

Figure 3. Boxplot of monthly charges by churn status.

Step 5: Split the Data into Training and Testing Sets

To evaluate model performance fairly, the dataset is divided into training and testing sets:

- The feature set (**X**) contains all columns except the target column `'Churn'`.
- The target variable (**y**) contains the `'Churn'` column indicating whether a customer left or stayed.

The `train_test_split()` function is used to split the data:

- **80%** of the data is used to train the models.
- **20%** is reserved for testing how well the models perform on unseen data.
- A `random_state` value is set to ensure reproducibility of the results.

In [8]:
```
# Step 5: Split the data into training and testing sets

# Separate features (X) and target (y)
X = df.drop('Churn', axis=1)
y = df['Churn']

# 80% training and 20% testing split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=42)
```

Step 6: Feature Scaling

Feature scaling is applied to ensure that all numerical features contribute equally to the model, especially for algorithms like **Logistic Regression** and **k-Nearest Neighbors (k-NN)** which are sensitive to feature magnitudes.

- `StandardScaler` from `scikit-learn` is used to standardize the features by removing the mean and scaling to unit variance.
- The scaler is fitted on the training set and then applied to both the training and test sets.
- This ensures that the model is trained on scaled values and avoids data leakage from the test set during scaling.

In [9]:
```
# Step 6: Feature Scaling
# Standardize features for models like k-NN and Logistic Regression
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 7: Train the Classification Models

In this step, three classification models are trained to predict customer churn:

- **Logistic Regression**
  A linear model used for binary classification. It is trained on the scaled training data and makes predictions on the scaled test data.
- **k-Nearest Neighbors (k-NN)**
  A distance-based model that classifies new data points based on the majority class of their nearest neighbors. Here, `k` is set to 5. This model also uses the scaled feature set.
- **Decision Tree Classifier**
  A tree-based model that splits the data into subsets using feature-based rules. Unlike the previous two models, it does not require feature scaling.

Each model is trained using the training data and then used to make predictions on the test data.

In [10]:
```
# Step 7: Train the Classification Models

# Logistic Regression
log_model = LogisticRegression()
log_model.fit(X_train_scaled, y_train)
log_preds = log_model.predict(X_test_scaled)

# k-NN
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_scaled, y_train)
knn_preds = knn_model.predict(X_test_scaled)

# Decision Tree (scaling not needed)
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
tree_preds = tree_model.predict(X_test)
```

Step 8: Evaluate All Models

This step evaluates the performance of each classification model using the `classification_report` from `scikit-learn`.

The report includes key metrics:

- **Accuracy**: Overall correctness of the model.
- **Precision**: Proportion of correctly predicted churners out of all predicted churners.
- **Recall**: Proportion of actual churners that were correctly identified.
- **F1-Score**: The harmonic mean of precision and recall, balancing both concerns.

Evaluation is performed on the test set predictions for:

- Logistic Regression
- k-Nearest Neighbors (k-NN)
- Decision Tree Classifier

These results help compare how well each model handles the churn prediction task.

In [11]:
```
# Step 8: Evaluate All Models

print("Logistic Regression:\n", classification_report(y_test, log_preds))
print("k-NN:\n", classification_report(y_test, knn_preds))
print("Decision Tree:\n", classification_report(y_test, tree_preds))
```

```
Logistic Regression:
               precision    recall  f1-score   support

           0       0.83      0.89      0.86      1033
           1       0.62      0.49      0.55       374

    accuracy                           0.79      1407
   macro avg       0.73      0.69      0.70      1407
weighted avg       0.77      0.79      0.78      1407

k-NN:
               precision    recall  f1-score   support

           0       0.82      0.82      0.82      1033
           1       0.51      0.51      0.51       374

    accuracy                           0.74      1407
   macro avg       0.67      0.67      0.67      1407
weighted avg       0.74      0.74      0.74      1407

Decision Tree:
               precision    recall  f1-score   support

           0       0.82      0.80      0.81      1033
           1       0.48      0.52      0.50       374

    accuracy                           0.72      1407
   macro avg       0.65      0.66      0.66      1407
weighted avg       0.73      0.72      0.73      1407
```

Step 9: Visual Summary

This section provides a visual summary of the classification analysis, helping to better understand class imbalance, feature relationships, and model performance.

- **Figure 5: Churn Class Distribution**
  A countplot showing the distribution of the target variable, Churn. It highlights the imbalance between churned and non-churned customers.
- **Figure 6: Correlation Heatmap**
  A heatmap displaying the pairwise correlation between numerical features. This helps identify relationships or multicollinearity in the dataset.
- **Figure 7: F1-Score Comparison**
  A bar chart comparing the F1-scores of the three classification models—Logistic Regression, k-NN, and Decision Tree. This metric provides a balanced view of model performance on the minority churn class.
- **Figures 8, 9, and 10: Confusion Matrices**
  Each figure displays a confusion matrix for one of the models. These visualizations reveal how many churn and non-churn predictions were correct or incorrect, offering insight into precision, recall, and misclassification rates.

In [21]:
```
# Step 9: Visual Summary (APA format captions and figure numbers)

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import f1_score, confusion_matrix, ConfusionMatrixDisplay

# 9.1 Churn Class Distribution (Figure 5)
fig, ax = plt.subplots()
sns.countplot(x=y, ax=ax)
ax.set_title("Churn Distribution")
ax.set_xlabel("Churn (0 = No, 1 = Yes)")
ax.set_ylabel("Count")
fig.subplots_adjust(bottom=0.25)
fig.text(0.5, 0.05,
         "Figure 5. Class distribution of the target variable Churn.",
         ha='center', fontsize=10)
plt.show()

# 9.2 Correlation Heatmap (Figure 6)
fig, ax = plt.subplots(figsize=(12, 10))
sns.heatmap(df.corr(), cmap='coolwarm', annot=False, ax=ax)
ax.set_title("Feature Correlation Heatmap")
fig.subplots_adjust(bottom=0.2)
fig.text(0.5, 0.05,
         "Figure 6. Heatmap showing pairwise correlation between all feature
s.",
         ha='center', fontsize=10)
plt.show()

# 9.3 F1 Score Comparison (Figure 7)
f1_scores = {
    "Logistic Regression": f1_score(y_test, log_preds),
    "k-NN": f1_score(y_test, knn_preds),
    "Decision Tree": f1_score(y_test, tree_preds)
}
fig, ax = plt.subplots()
sns.barplot(x=list(f1_scores.keys()), y=list(f1_scores.values()), ax=ax)
ax.set_title("F1-Score Comparison")
ax.set_ylabel("F1 Score")
ax.set_ylim(0, 1)
fig.subplots_adjust(bottom=0.25)
fig.text(0.5, 0.05,
         "Figure 7. F1-score comparison of Logistic Regression, k-NN, and Deci
sion Tree models.",
         ha='center', fontsize=10)
plt.show()

# 9.4 Confusion Matrices (Figures 8, 9, 10)
models = {
    'Logistic Regression': log_preds,
    'k-NN': knn_preds,
    'Decision Tree': tree_preds
}
fig_num = 8
for name, preds in models.items():
    cm = confusion_matrix(y_test, preds)
    fig, ax = plt.subplots()
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])
    disp.plot(cmap='Blues', ax=ax)
    plt.title(f"Confusion Matrix: {name}")
    fig.subplots_adjust(bottom=0.25)
    fig.text(0.5, 0.05,
             f"Figure {fig_num}. Confusion matrix for {name} model predictions
on test data.",
             ha='center', fontsize=10)
    fig_num += 1
    plt.show()
```

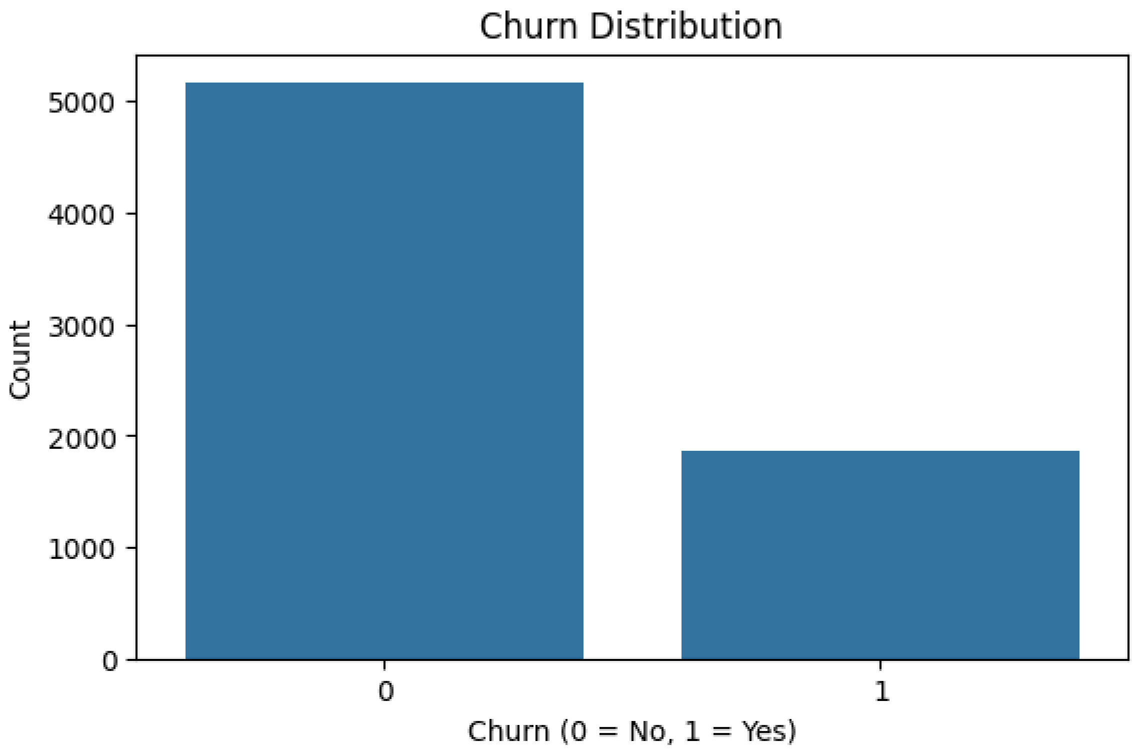

Figure 5. Class distribution of the target variable Churn.

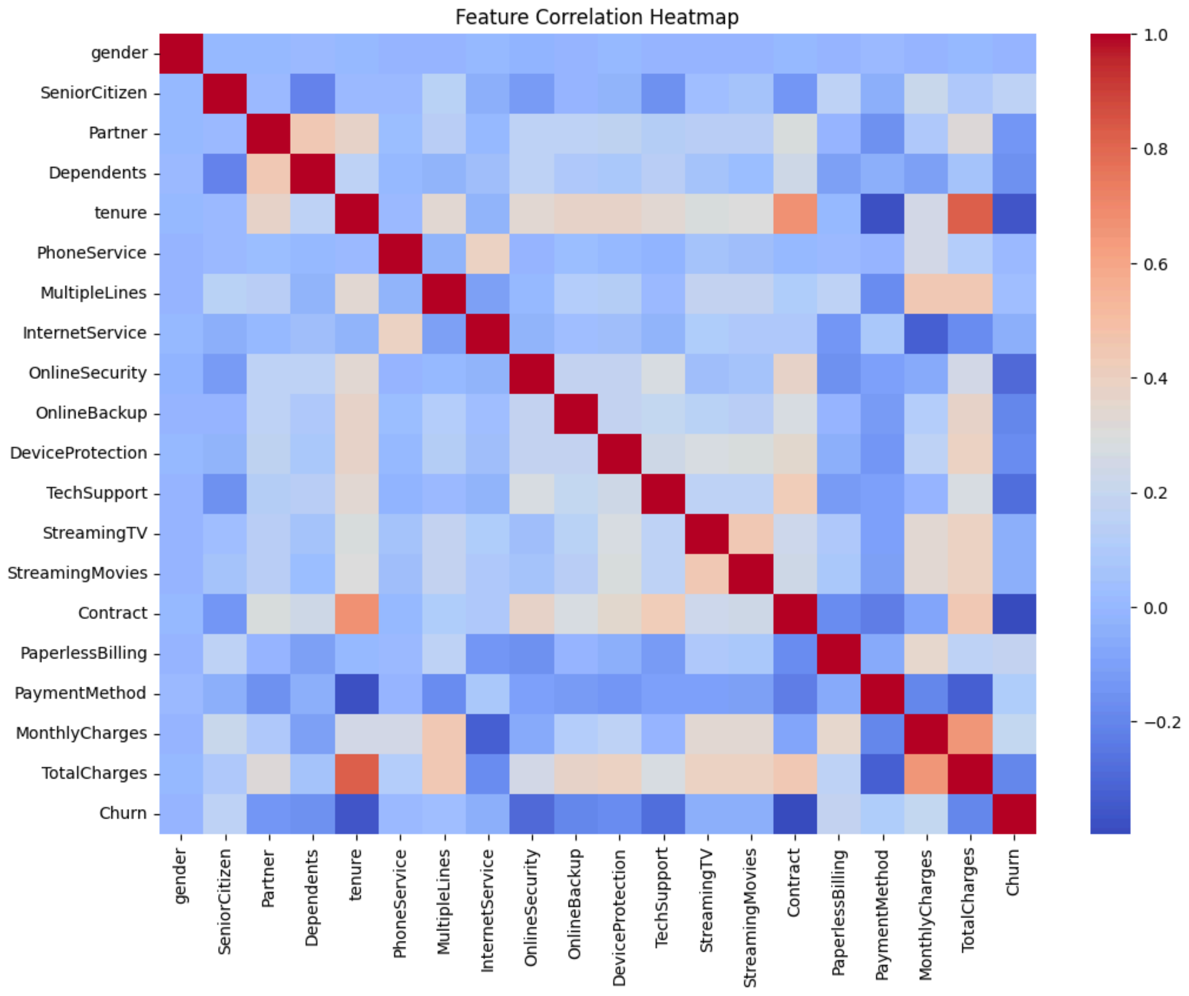

Figure 6. Heatmap showing pairwise correlation between all features.

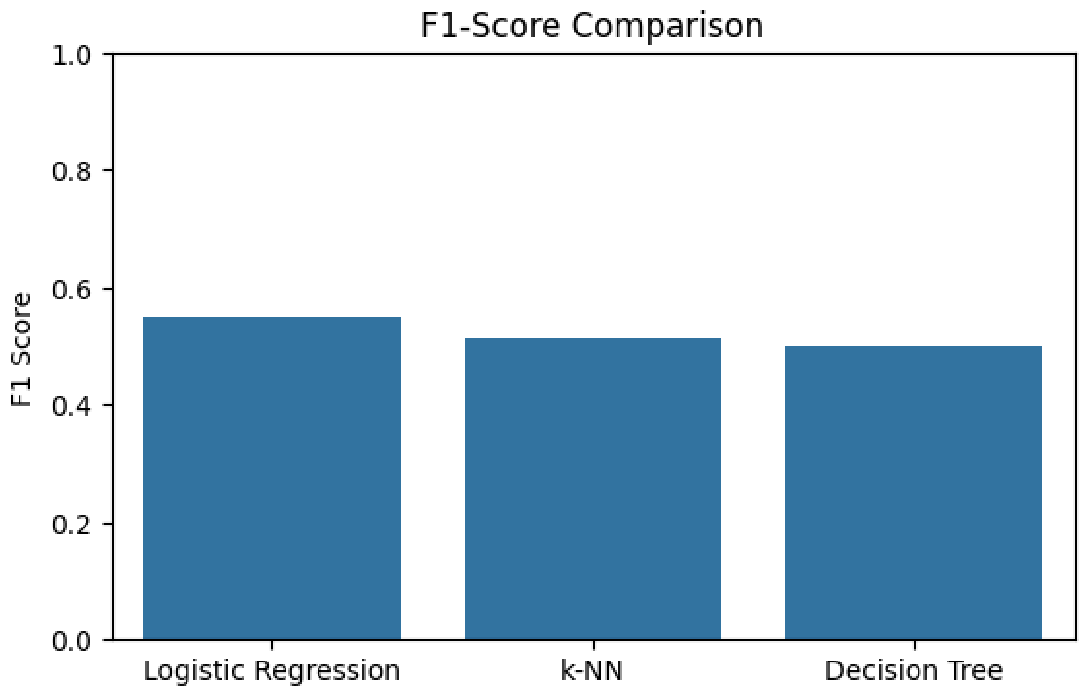

Figure 7. F1-score comparison of Logistic Regression, k-NN, and Decision Tree models.

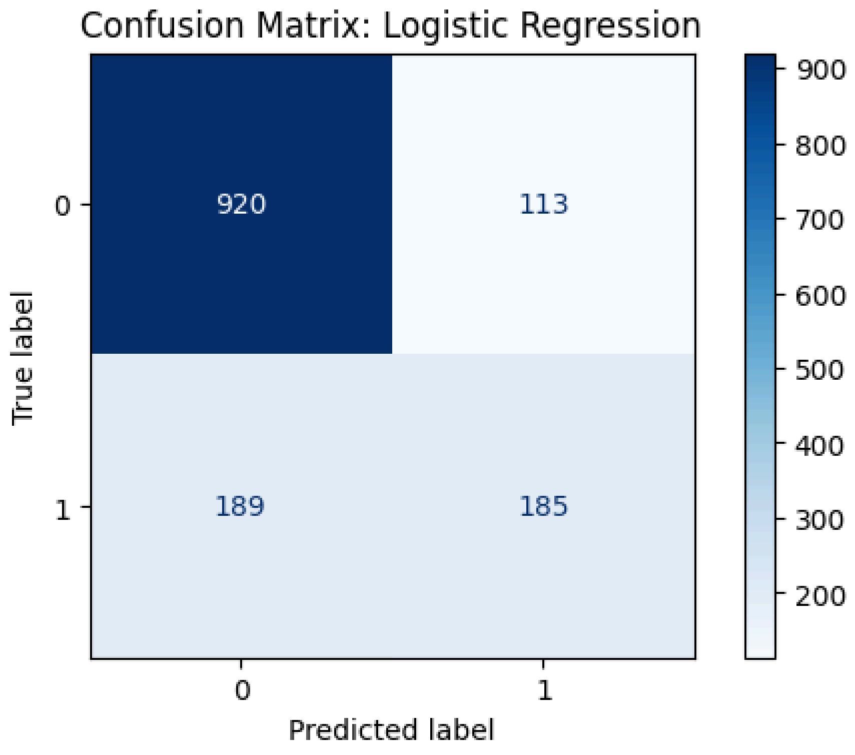

Figure 8. Confusion matrix for Logistic Regression model predictions on test data.

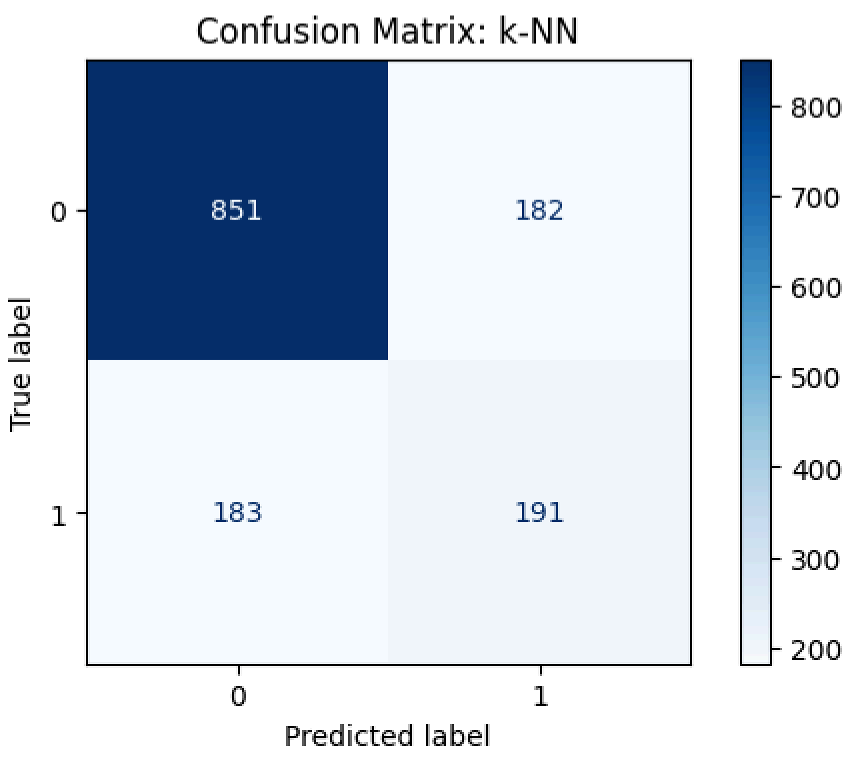

Figure 9. Confusion matrix for k-NN model predictions on test data.

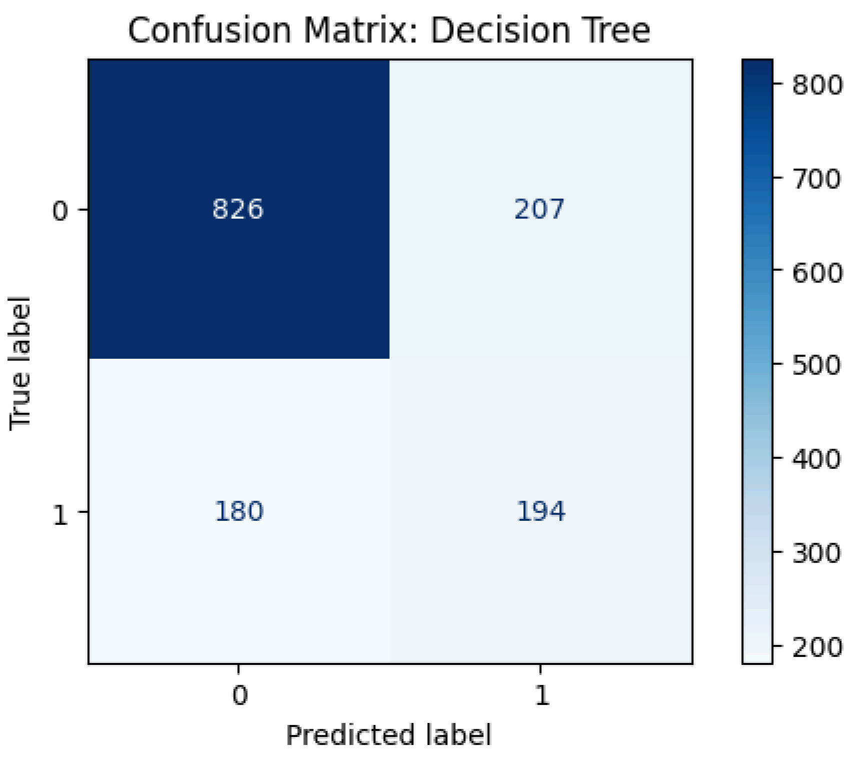

Figure 10. Confusion matrix for Decision Tree model predictions on test data.

Model Development and Evaluation Metrics :

I trained and tested three models: Logistic Regression, k-Nearest Neighbors (k=5), and Decision Tree Classifier. In all cases, models were trained on the training data set and applied to the same test data set for concordance.

The models were tested in terms of: • Precision is the fraction of true hits among all predicted positive instances. • Precision: This measure tells us what proportion of customers that we targeted to churn cases did churn. • Recalling is checking how well the model identifies actual churn customers. • F1-score integrates both precision and recall into a single metric, which is critical when the number of negative examples is largely unbalanced to the positive ones.

To gain deeper insight about the correct vs. incorrect prediction distribution, confusion matrices were consulted as well (Figures 8–10). A correlation heatmap (Fig. 6) and class distribution chart (Fig. 5) were useful to interpret the feature-target relationships. Furthermore, Figure 7 visualizes the spread of F1-scores for all implemented models.

Discussion :

In general, the best performance was given by the Logistic Regression model. It had 79 percent accuracy and an F1-score of 0.55, so it did a good job in correctly identifying both who stayed and who left. Its precision and recall were close to each other, indicating that it was not biased toward either class. It also made fewer errors compared to the other models as evidenced in confusion matrix (Figure 8). Due to its stable and reliable performance, Logistic Regression was the most robust model in present research.

The k-Nearest Neighbors (k-NN) model resulted in 74% accuracy and an F1-score of 0.51. It was significantly more sensitive to problems such as not carrying the same number of samples in each class and the scale of the data. This had the effect of making it more difficult for the model to distinguish customers who would churn. The Decision Tree model had the highest recall (0.52), which implies this model was the most capable of finding churned customers. But it was less precise (0.48), so it also incorrectly guessed that that many loyal customers would leave. This could result in ill-advised responses from the business. The confusion matrices in Figures 9 and 10, as well as class imbalance in Fig. 5, show that it is imperative to consider other metrics than accuracy when evaluating models.

Conclusion :

In this paper, the performance of three machine learning methods-Logistic Regression, k-NN, and Decision Tree were compared in terms of customer churn prediction based on the Telco dataset. All models were constructed with Python’s scikit-learn library and conforming the pre-processing and evaluation procedures.

Logistic Regression is the best performing method out of three, with it a good balance of precision and recall as well as computationally fast. The Decision Tree model proved to be promising with great recall, which will be ideal for businesses where missed churners are considered more deleterious than false positives. k-NN, despite being the simplest of architecture, had limitations given the features of the present dataset.

For future development, we suggest exploring ensemble models such as RandomForest or XGBoost and class imbalance corrective methods such as SMOTE to mitigate the effects of the negative class imbalance. The robustness of the model may be increased by adding the time-series features or behavioral data. By optimizing these models, telecom operators can proactively predict and retain customers before they are churning and save a significant amount on churn-related losses.

Step 10: Overall

Model Comparison & Business Insight

Metric	Logistic Regression	k-NN	Decision Tree
Accuracy	0.79	0.74	0.72
Precision (Churn=1)	0.62	0.51	0.48
Recall (Churn=1)	0.49	0.51	0.52
F1-Score (Churn=1)	0.55	0.51	0.50

Observations

- **Logistic Regression** has the highest overall accuracy and F1-score.
- **k-NN** is simple but needs proper scaling.
- **Decision Tree** has good recall but may overfit.

Recommendation

- Use Logistic Regression as a baseline.
- Consider ensemble methods or SMOTE for improvement.

References

Ahmed, S., Ameen, A., & Rizwan, M. (2016). Predicting Customer Churn in Telecom Industry Using Multilayer Perceptron Neural Networks: A Case of Ethiopia. International Journal of Computer Applications, 975, 8887.

Han, J., Kamber, M., & Pei, J. (2011). Data mining: concepts and techniques. Elsevier.

Hadden, J., Tiwari, A., Roy, R., & Ruta, D. (2007). Computer assisted customer churn management: State-of-the-art and future trends. Computers & Operations Research, 34(10), 2902–2917.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

GitHub Link : https://github.com/srinivasbhootam/customer-churn-prediction-ml* (https://github.com/srinivasbhootam/customer-churn-prediction-ml*)