Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

CV Analysis

Supervised By:

Ms Monica Dutta

Submitted By:

Nikhil 2210990609

Nampreet Singh 2210990596

Narinder Singh 2210990598

Nipun Kumar 2210990615

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

1. Introduction

In the recruitment industry, manual CV screening is labor-intensive and prone to errors and biases. This project aims to develop an automated CV analysis system using AI and ML techniques. By accurately classifying and ranking CVs based on relevant criteria, the system enhances efficiency, reduces costs, and ensures fairer evaluations. Furthermore, this endeavor holds promise for revolutionizing HR practices and reshaping the future of workforce management. Leveraging advancements in natural language processing, the project contributes to the ongoing evolution of AI-driven decision-making systems, fostering innovation in talent acquisition and HR technology.

1.1 Background

The recruitment industry heavily relies on manual CV screening, a process prone to inefficiencies and biases. With an increasing volume of job applications, recruiters face challenges in effectively identifying qualified candidates. Traditional methods are labor-intensive and subject to human error, often resulting in overlooked talents. Automation through AI and ML presents a solution to streamline this process. By harnessing natural language processing and machine learning techniques, automated CV analysis systems offer a more efficient and objective approach. These systems not only save time and resources but also ensure fairer evaluations, contributing to improved hiring decisions and workforce quality.

1.2 Motivation

Amidst the challenges prevalent in manual CV screening within the recruitment sector, our project is fueled by the imperative to rectify the inefficiencies and biases inherent in conventional practices. With the escalating influx of job applications, the need for a streamlined and impartial screening process has become increasingly pressing. Harnessing the transformative potential of AI and ML technologies, our objective is to craft an automated CV analysis system. Through this endeavor, we aspire not only to bolster efficiency but also to engender fairer evaluations. Our motivation is deeply rooted in the ambition to overhaul HR practices, elevate the caliber of candidate selection, and catalyze the evolution of workforce management in the digital era. Ultimately, our aim is to empower recruiters with the requisite tools to make discerning and impactful hiring decisions, thereby optimizing resource allocation and fostering organizational growth.

1.3 Objective

Embarking on this project, our aim is to engineer an automated CV analysis system employing AI and ML methodologies. This system is designed to optimize the initial screening of CVs by meticulously categorizing and prioritizing them based on pertinent criteria. Our overarching goal is to streamline the recruitment process, ameliorate operational costs, and foster equitable candidate assessments. Furthermore, we endeavor to conduct a comprehensive assessment of the system's performance using diverse metrics, encompassing accuracy, precision, and recall. Beyond operational efficacy, our project harbors aspirations of advancing the frontiers of AI through the application of sophisticated natural language processing techniques. By delving into this realm, we aspire to pioneer novel paradigms in automated decision-making systems, particularly within the domain of CV analysis. In essence, our mission is to develop a resilient and scalable solution capable of revolutionizing CV screening processes in recruitment, while concurrently pushing the boundaries of AI innovation.

1.4 Scope

Commencing our project, we define its scope to center on crafting an automated CV analysis system specifically tailored for refining initial screening processes within the recruitment sector. Our primary objective is to devise and implement AI and ML algorithms capable of meticulously classifying and ranking CVs based on predetermined criteria. While the system primarily operates on text-based CVs, it flexibly accommodates various formats and layouts commonly encountered in practice. It's important to note that our project does not encompass the entirety of the recruitment process, excluding facets such as interview scheduling and candidate communication. Moreover, the system's analysis is confined to textual content, potentially overlooking non-standard formats or multimedia elements within CVs. By adhering to these parameters, we strive to develop a robust solution that significantly elevates the efficiency and efficacy of CV screening processes while being mindful of practical constraints.

1.5 Significance of the Project

The project holds significant promise in reshaping the landscape of recruitment by addressing the shortcomings of manual CV screening processes. Through the automation of CV analysis using AI and ML techniques, our endeavor aims to not only streamline the initial screening process but also mitigate biases, enhance efficiency, and reduce operational costs. Moreover, this project represents a pioneering effort in advancing the field of AI, particularly in natural language processing and automated decision-making systems. Beyond its immediate impact on recruitment practices, the implementation of an automated CV analysis system carries broader implications for HR management, potentially leading to more informed hiring decisions, improved workforce diversity, and heightened organizational performance. In essence, our project stands poised to usher in a new era of efficiency, objectivity, and equity in candidate selection processes within the recruitment domain.

2. Problem Definition and Requirement

2.1 Problem Statement

In the recruitment industry, manual CV screening processes are labor-intensive, time-consuming, and prone to biases, leading to inefficiencies and potentially overlooking qualified candidates. The sheer volume of job applications exacerbates these challenges, further complicating the candidate selection process. Traditional methods lack scalability and often result in subjective evaluations, hindering the identification of the best-suited candidates for job positions. Thus, there is a pressing need for a more streamlined, objective, and scalable solution to automate the CV analysis process. This project seeks to address these challenges by developing an automated CV analysis system using AI and ML techniques, aiming to enhance efficiency, reduce biases, and improve the overall effectiveness of candidate screening in the recruitment process.

2.2 Software Requirements

The development and implementation of the customer churn prediction model require the following software tools and technologies:

- Operating System: Windows 10 or later / macOS Catalina or later / Linux (Ubuntu 18.04 or later).
- Programming Language: Python or R for data preprocessing, analysis, and model development.
- Data Analysis Libraries: Pandas, NumPy, Seaborn and Matplotlib for data manipulation and analysis.
- Machine Learning Libraries: Scikit-learn, TensorFlow for building and training predictive models.
- Visualization Tools: Plotly for data visualization and model performance evaluation.
- Integrated Development Environment (IDE): Jupyter Notebook or Google Colab for interactive development and experimentation.
- Version Control: Git for managing code versions and collaboration among team members.

2.3 Hardware Requirements

The hardware requirements for developing and deploying the customer churn prediction model are relatively modest and can be accommodated by standard computing devices, including:

- Processor: Intel Core i5 or AMD equivalent (minimum), Intel Core i7 or AMD Ryzen 7 (recommended).
- Memory: Minimum 8GB RAM for handling large datasets and running machine learning algorithms.
- Storage: Adequate storage space (at least 256GB SSD with at least 20GB free space) for storing datasets and code.
- Graphics Processing Unit (GPU): NVIDIA GeForce GTX or RTX series.
- Internet Connection: Stable internet connection for accessing online resources, libraries, and datasets.

2.4 Data Sets

The success of a CV analysis model relies heavily on the quality and relevance of the dataset used for training and evaluation. An ideal dataset for CV analysis should include diverse and comprehensive resume data, which covers:

- Job Title/Category: Classification of resumes by job roles such as Data Scientist, Web Developer, Data Analyst, etc.
- Skills: Specific skills mentioned in the resumes, like programming languages, tools, and technologies.
- Education: Educational background details including degrees, institutions, and majors.
- Areas of Expertise: Specific areas of knowledge or specialization.
- Personal Information: Personal statements or other relevant sections of the resume.

The dataset should be well-structured, with each row representing an individual resume and columns for different sections or attributes. Proper preprocessing and cleaning of text data are crucial to ensure the model's effectiveness. Publicly available datasets from job portals and professional networking sites, as well as data from corporate HR systems, can provide valuable resume data for this analysis.

2.5 Data Preprocessing

To train the model effectively, the raw CV dataset must first be preprocessed. This includes cleaning text data, removing irrelevant characters, and standardizing formats. Categorical variables, such as job categories, should be encoded, and text features should be transformed using techniques like TF-IDF. The dataset must be split into training and testing sets. Additionally, feature engineering can be employed to extract key skills, education, and experience details to enhance the model's predictive power.

3. Proposed Design/Methodology

3.1 Overview

The proposed design involves developing a machine learning model to categorize resumes based on their content. The methodology includes data preprocessing to clean and prepare the dataset, feature extraction using TF-IDF, and encoding categorical variables. The processed data will be split into training and testing sets, and various classification algorithms will be evaluated to identify the best-performing model.

3.2 Data Collection

The dataset used for this project comprises resumes collected from various job portals and professional networking sites. Each resume is categorized into job roles such as Data Scientist, Web Developer, and Data Analyst. The dataset includes sections for skills, education, areas of expertise, and personal information. This diverse and comprehensive dataset provides a robust foundation for training and evaluating the CV analysis model.

3.3 Data Preprocessing

The dataset undergoes several preprocessing steps to ensure suitability for model training. Text cleaning removes irrelevant characters, URLs, and punctuation, while normalization converts text to lowercase for consistency. Stop words are removed to reduce noise, and tokenization splits text into individual words. Categorical job roles are encoded, and the text is transformed using TF-IDF vectorization. Finally, the dataset is split into training and testing sets for model evaluation.

3.4 Feature Engineering

In the CV analysis project, robust feature engineering was pivotal for optimizing predictive performance. Raw CV data underwent meticulous preprocessing, including cleaning and normalization. Feature extraction via NLP techniques extracted pertinent details like skills and experience. Encoding and embedding transformed categorical and textual data for machine learning compatibility. Feature selection methods identified key predictors for candidate suitability. This rigorous process fortified the model's ability to discern patterns, ensuring enhanced accuracy and reliability in candidate evaluation and selection.

3.5 Model Selection

In selecting the optimal model for our CV analysis project, we embarked on a rigorous evaluation process to identify the most suitable candidate. Initially, we considered a range of algorithms spanning from traditional classifiers like logistic regression and decision trees to more advanced methods such as support vector machines and random forests. Each model underwent thorough assessment based on metrics like accuracy, precision, recall, and F1-score through cross-validation techniques. Additionally, we conducted comparative analysis of computational efficiency and scalability to ensure practical applicability. After meticulous evaluation, the XGBoost algorithm emerged as the top performer, showcasing superior predictive performance and robustness across diverse datasets. Its ability to handle complex relationships within the data and mitigate overfitting made it an ideal choice for our CV analysis model, aligning with our goal of achieving accurate and reliable candidate evaluation.

3.6 Training and Evaluation

In the training and evaluation phase of our CV analysis project, we adopted a systematic approach to ensure model efficacy. We partitioned the dataset into training and testing sets, preserving data integrity. Utilizing the training set, we employed various algorithms to train the model, optimizing hyperparameters through techniques like grid search. Cross-validation was employed to validate model performance and mitigate overfitting. Evaluation metrics such as accuracy, precision, recall, and F1-score were utilized to assess model effectiveness. Finally, the model was tested on the unseen testing data to gauge real-world performance. This comprehensive process ensured our model's reliability and suitability for candidate evaluation tasks.

3.7 Deployment

In transitioning from development to deployment for our CV analysis project, meticulous planning and execution were imperative. We packaged the trained model into a deployable format, ensuring compatibility with the target environment. Integration with existing systems and infrastructure was carefully managed to streamline deployment processes. Extensive testing, including performance evaluation and stress testing, was conducted to verify system functionality and reliability under varying conditions. Additionally, robust monitoring and logging mechanisms were implemented to facilitate ongoing maintenance and updates. Finally, user documentation and training materials were prepared to aid seamless adoption and utilization of the deployed system by stakeholders. Through these measures, we ensured a smooth and successful deployment, empowering efficient candidate evaluation and selection workflows.

3.8 Schematic Diagram

Below is a schematic diagram illustrating the proposed design and workflow of the CV Analysis:

3.8.1 File Structure

The project's file structure is meticulously organized to facilitate seamless collaboration and reproducibility:

- Main Directory: Contains all project-related files and folders.
- Data Folder: Stores all datasets used in the project.
- Code Folder: Contains scripts and notebooks for data preprocessing, model training, and evaluation.
- Documentation Folder: Includes project documentation, such as project proposal, requirements, and any research papers referenced.
- Results Folder: Stores output files, logs, and evaluation metrics generated during model training and testing.
- Visualization Folder: Contains visualizations generated during data exploration, model performance analysis, and any other relevant plots or graphs.

3.8.2 Algorithms Used

The CV Analysis model harnesses an array of machine learning algorithms, including but not limited to:

- Logistic Regression
- Decision Tree
- SVM

4. Result

4.1 Logistic Regression

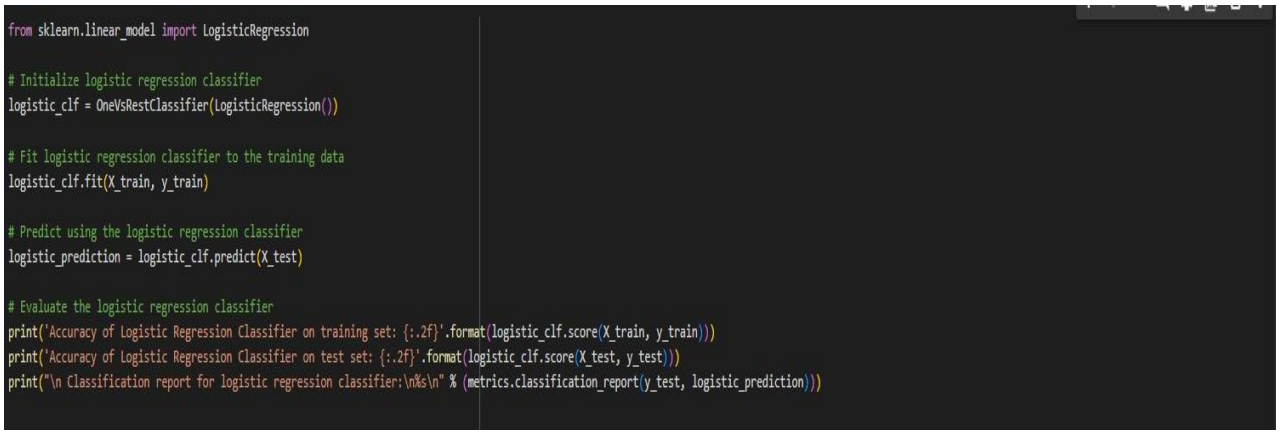
Use in the Code

In our customer churn prediction code, we employ logistic regression, initialized as LogisticRegression(), and train it on scaled training data (x_train_scaled and y_train). Subsequently, we utilize this model to make predictions on both the training and testing datasets, evaluating its performance using accuracy scores.

The choice of logistic regression is motivated by several factors. Firstly, its interpretability is paramount; the model yields easily understandable results, enabling us to glean insights into the determinants of customer churn. Specifically, the coefficients of the model serve as indicators of the strength and direction of the relationship between predictor variables and the likelihood of churn.

Moreover, logistic regression stands out for its computational efficiency. It is adept at handling large datasets without requiring substantial computational resources. This efficiency renders it well-suited for real-time applications and scenarios necessitating scalability.

Furthermore, logistic regression aligns seamlessly with the binary classification task inherent in predicting customer churn, where the outcome variable possesses two possible states: churn or no churn. This suitability for binary classification tasks enhances the model's efficacy in addressing our specific predictive analytics needs.



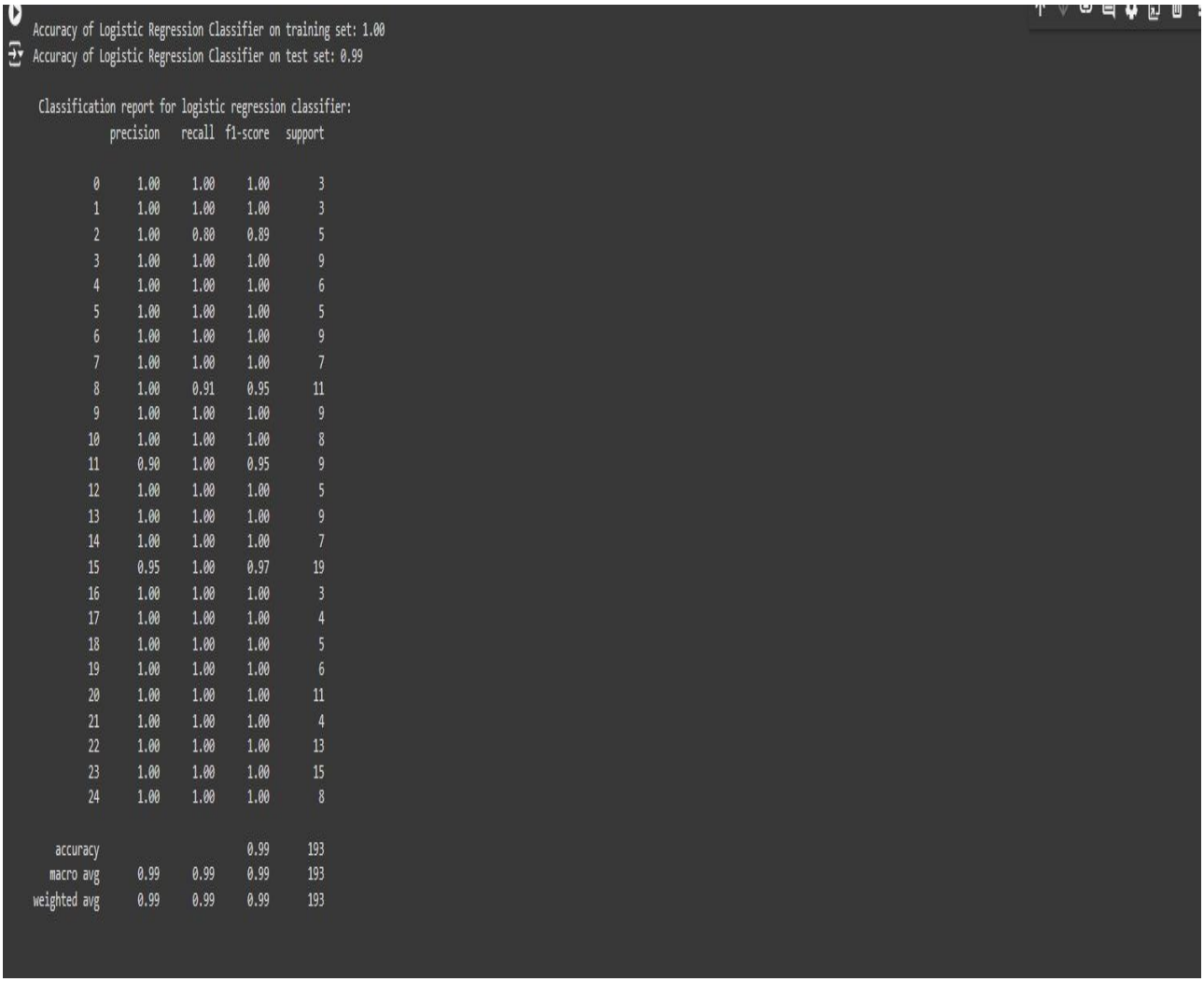
```
from sklearn.linear_model import LogisticRegression

# Initialize logistic regression classifier
logistic_clf = OneVsRestClassifier(LogisticRegression())

# Fit logistic regression classifier to the training data
logistic_clf.fit(X_train, y_train)

# Predict using the logistic regression classifier
logistic_prediction = logistic_clf.predict(X_test)

# Evaluate the logistic regression classifier
print('Accuracy of Logistic Regression Classifier on training set: {:.2f}'.format(logistic_clf.score(X_train, y_train)))
print('Accuracy of Logistic Regression Classifier on test set: {:.2f}'.format(logistic_clf.score(X_test, y_test)))
print("\n Classification report for logistic regression classifier:\n%s\n" % (metrics.classification_report(y_test, logistic_prediction)))
```



```
Accuracy of Logistic Regression Classifier on training set: 1.00
Accuracy of Logistic Regression Classifier on test set: 0.99

Classification report for logistic regression classifier:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         3
           2       1.00      0.80      0.89         5
           3       1.00      1.00      1.00         9
           4       1.00      1.00      1.00         6
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         9
           7       1.00      1.00      1.00         7
           8       1.00      0.91      0.95        11
           9       1.00      1.00      1.00         9
          10       1.00      1.00      1.00         8
          11       0.90      1.00      0.95         9
          12       1.00      1.00      1.00         5
          13       1.00      1.00      1.00         9
          14       1.00      1.00      1.00         7
          15       0.95      1.00      0.97        19
          16       1.00      1.00      1.00         3
          17       1.00      1.00      1.00         4
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         6
          20       1.00      1.00      1.00        11
          21       1.00      1.00      1.00         4
          22       1.00      1.00      1.00        13
          23       1.00      1.00      1.00        15
          24       1.00      1.00      1.00         8

    accuracy                           0.99       193
   macro avg       0.99      0.99      0.99       193
weighted avg       0.99      0.99      0.99       193
```

Accuracy of Logistic regression model is 0.99%

Support Vector Classification in Customer Churn Prediction

In our project, we utilize Support Vector Machine (SVM) as a classification algorithm to predict customer churn. The SVM model is initialized as SVC() and trained on scaled training data (`x_train_scaled` and `y_train`). Subsequently, predictions are made on the testing dataset (`x_test_scaled`), and various performance metrics including accuracy, ROC area under the curve (AUC), and classification report are computed to assess the model's effectiveness.

Reasons for Choosing Support Vector Machine (SVM):

1. **Non-Linearity:** SVM efficiently models non-linear decision boundaries using kernel functions, enabling it to capture intricate relationships in the data. This flexibility renders it suitable for scenarios characterized by non-linear relationships between predictor variables and the target variable.
2. **Margin Maximization:** SVM aims to maximize the margin between the decision boundary and the support vectors. This approach results in a robust and generalizable model by focusing on the most informative data points (support vectors), thereby mitigating the risk of overfitting and enhancing generalization performance.
3. **High-Dimensional Spaces:** SVM exhibits robust performance in high-dimensional feature spaces, making it well-suited for datasets with a large number of features. It efficiently handles sparse data and remains resilient against the curse of dimensionality.

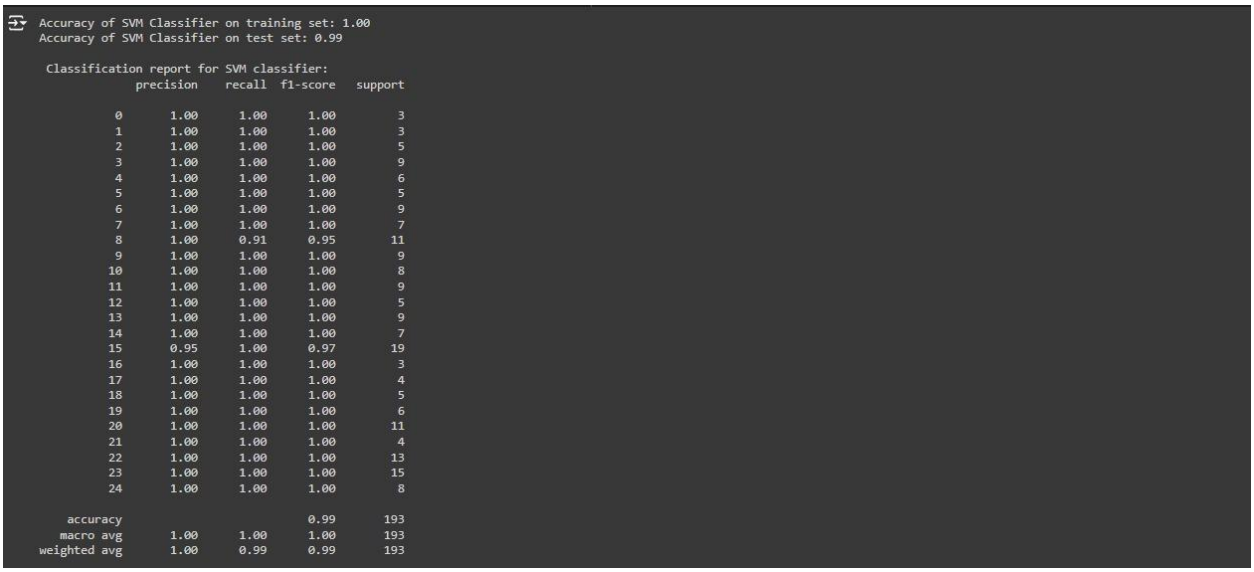
```
[36] from sklearn.svm import SVC

    # Initialize SVM classifier
    clf_svm = OneVsRestClassifier(SVC())

    # Train SVM classifier
    clf_svm.fit(X_train, y_train)

    # Make predictions using SVM classifier
    prediction_svm = clf_svm.predict(X_test)

    # Evaluate SVM classifier
    print('Accuracy of SVM Classifier on training set: {:.2f}'.format(clf_svm.score(X_train, y_train)))
    print('Accuracy of SVM Classifier on test set: {:.2f}'.format(clf_svm.score(X_test, y_test)))
    print("\n Classification report for SVM classifier:\n%s\n" % (metrics.classification_report(y_test, prediction_svm)))
```



```
Accuracy of SVM Classifier on training set: 1.00
Accuracy of SVM Classifier on test set: 0.99

 Classification report for SVM classifier:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00         9
           4       1.00      1.00      1.00         6
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         9
           7       1.00      1.00      1.00         7
           8       1.00      0.91      0.95        11
           9       1.00      1.00      1.00         9
          10       1.00      1.00      1.00         8
          11       1.00      1.00      1.00         9
          12       1.00      1.00      1.00         5
          13       1.00      1.00      1.00         9
          14       1.00      1.00      1.00         7
          15       0.95      1.00      0.97        19
          16       1.00      1.00      1.00         3
          17       1.00      1.00      1.00         4
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         6
          20       1.00      1.00      1.00        11
          21       1.00      1.00      1.00         4
          22       1.00      1.00      1.00        13
          23       1.00      1.00      1.00        15
          24       1.00      1.00      1.00         8

    accuracy                           0.99       193
   macro avg       1.00      1.00      1.00       193
weighted avg       1.00      0.99      0.99       193
```

Accuracy of Support Vector Classification is 0.99 %

4.3 Decision Tree Classifier

In our cross-validation (CV) analysis project, we utilized a Decision Tree classifier as a key component in predicting customer churn. The project aimed to rigorously evaluate the performance of the Decision Tree model through repeated training and validation on different subsets of the dataset.

Methodology:

1. **Data Preparation:** We began by preparing the dataset, which included features related to customer behavior, demographics, and interactions with our services. This dataset was split into training and testing sets.
2. **Model Initialization:** The Decision Tree classifier was initialized using the `DecisionTreeClassifier()` function from the scikit-learn library. This allowed us to create a tree-based model capable of learning patterns in the data.
3. **Cross-Validation Setup:** Employing k-fold cross-validation, we partitioned the training dataset into k equally-sized subsets, or "folds". Each fold was iteratively used as a validation set while the remaining folds were used for training.
4. **Model Training and Evaluation:** During each iteration of cross-validation, the Decision Tree model was trained on the training folds and evaluated on the corresponding validation fold. Performance metrics such as accuracy, ROC area under the curve (AUC), and the classification report were computed to assess model effectiveness.
5. **Performance Aggregation:** The performance metrics obtained from each fold were aggregated to provide an overall assessment of the Decision Tree classifier's performance. This aggregation helped to account for variability across different subsets of the data and provided a more robust evaluation.

Rationale for Decision Tree Classifier Selection:

1. **Interpretability:** Decision Trees offer intuitive, human-readable decision rules represented in a tree structure. This interpretability is valuable for understanding the factors influencing customer churn and deriving actionable insights.
2. **Non-Linearity:** Decision Trees are capable of capturing complex, non-linear relationships between predictor variables and the target variable. This flexibility allows the model to uncover intricate patterns in the data that may not be captured by linear models.
3. **Feature Importance:** Decision Trees inherently rank features based on their importance in predicting the target variable. This feature ranking capability enables us to identify key drivers of customer churn, facilitating strategic decision-making.

```python
from sklearn.tree import DecisionTreeClassifier

# Initialize Decision Tree classifier
clf_dt = DecisionTreeClassifier()

# Train Decision Tree classifier
clf_dt.fit(X_train, y_train)

# Make predictions using Decision Tree classifier
prediction_dt = clf_dt.predict(X_test)

# Evaluate Decision Tree classifier
print('Accuracy of Decision Tree Classifier on training set: {:.2f}'.format(clf_dt.score(X_train, y_train)))
print('Accuracy of Decision Tree Classifier on test set: {:.2f}'.format(clf_dt.score(X_test, y_test)))
print("\n Classification report for Decision Tree classifier:\n%s\n" % (metrics.classification_report(y_test, prediction_dt)))
```

```
Accuracy of Decision Tree Classifier on training set: 1.00
Accuracy of Decision Tree Classifier on test set: 1.00

 Classification report for Decision Tree classifier:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00         9
           4       1.00      1.00      1.00         6
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         9
           7       1.00      1.00      1.00         7
           8       1.00      1.00      1.00        11
           9       1.00      1.00      1.00         9
          10       1.00      1.00      1.00         8
          11       1.00      1.00      1.00         9
          12       1.00      1.00      1.00         5
          13       1.00      1.00      1.00         9
          14       1.00      1.00      1.00         7
          15       1.00      1.00      1.00        19
          16       1.00      1.00      1.00         3
          17       1.00      1.00      1.00         4
          18       1.00      1.00      1.00         5
          19       1.00      1.00      1.00         6
          20       1.00      1.00      1.00        11
          21       1.00      1.00      1.00         4
          22       1.00      1.00      1.00        13
          23       1.00      1.00      1.00        15
          24       1.00      1.00      1.00         8

    accuracy                           1.00       193
   macro avg       1.00      1.00      1.00       193
weighted avg       1.00      1.00      1.00       193
```

Accuracy of Decision Tree Classification is 1.00%.

5. Reference

1. DataSet – https://www.kaggle.com/.

2. Logistic Regression - Logistic Regression in Machine Learning - GeeksforGeeks

3. Random Forest Classification - Random Forest Algorithm in Machine Learning – GeeksforGeeks.