CMPE 442 - Assignment 3

1. Problem Definition

In this project, we are studying with the data which is dedicated to classification problem related to the post-operative life expectancy in the lung cancer patients. Class 1 represents death within one year after surgery and class 2 represents survival. The purpose of the project is to analyze and process the data, then select the appropriate model for the data, finally fine-tune and test the model. To achieve this, we cleaned processed and scaled the raw data, then we tested the models that we thought would be suitable for the classification problem and determined the most suitable model.

2. Data Analysis

In this section, we analyzed the data by visualizing the features and creating some statistics. In our dataset, we have 17 features and last feature is representing the classes. 10 of these features are Boolean feature, 3 of them are numeric feature and 3 of them are categorical feature. Also, the number of instances is 470 and there are no missing values or missing cells.

The names of the properties, the values they take and what they represent are as follows:

Categorical Features

- DGN: Diagnosis - specific combination of ICD-10 codes for primary and secondary as well multiple tumours if any (DGN3, DGN2, DGN4, DGN6, DGN5, DGN8, DGN1)
- PRE6: Performance status - Zubrod scale (PRZ2, PRZ1, PRZ0)
- PRE14: T in clinical TNM - size of the original tumour, from OC11 (smallest) to OC14 (largest) (OC11, OC14, OC12, OC13)

Numeric Features

- PRE4: Forced vital capacity - FVC
- PRE5: Volume that has been exhaled at the end of the first second of forced expiration - FEV1
- AGE: Age at surgery

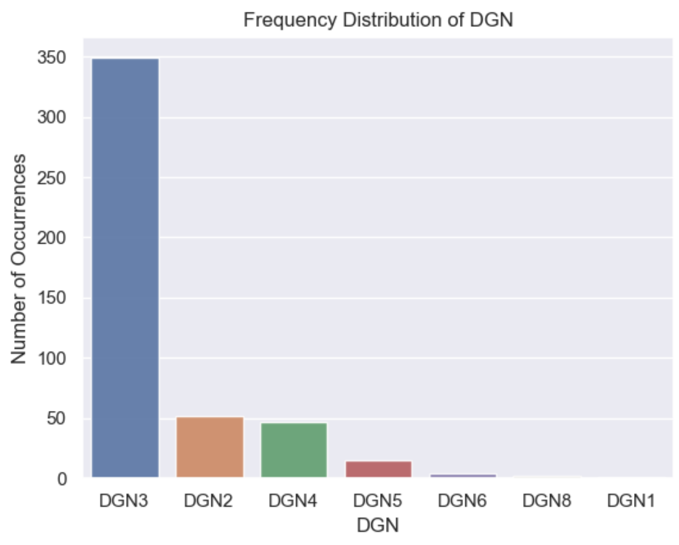
Binary Features (T/F)

- PRE7: Pain before surgery (T, F)
- PRE8: Haemoptysis before surgery (T, F)
- PRE9: Dyspnoea before surgery (T, F)
- PRE10: Cough before surgery (T, F)
- PRE11: Weakness before surgery (T, F)
- PRE17: Type 2 DM - diabetes mellitus (T, F)
- PRE19: MI up to 6 months (T, F)
- PRE25: PAD - peripheral arterial diseases (T, F)
- PRE30: Smoking (T, F)
- PRE32: Asthma (T, F)
- **Risk1Y**: 1 year survival period - (T)rue value if died (T, F) **(Class)**

To analyzing and visualizing the data we used pandas profiling library and created a report of our dataset. We also created some graphics for the features we want to examine.

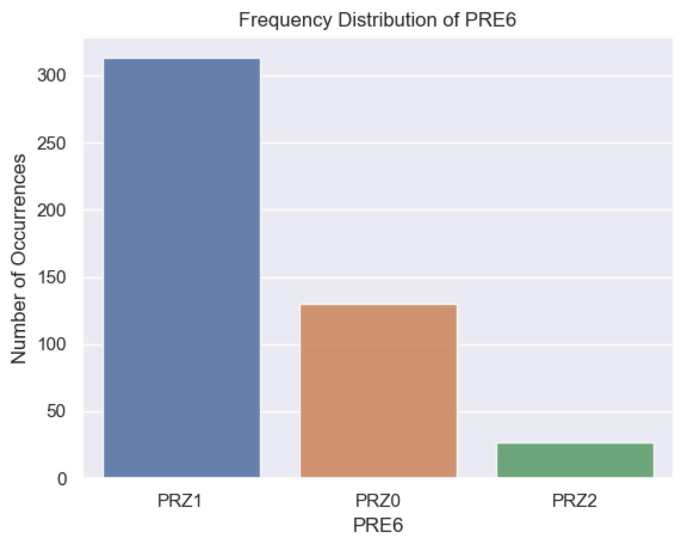
DGN

This is the frequency distribution of DGN. According to the graph, DGN3 appears 349 times, DGN2 appears 52 times, DGN4 appears 47 times, DGN5 appears 15 times, DGN6 appears 4 times, DGN8 appears 2 times and DGN1 appears 1 time in the dataset.



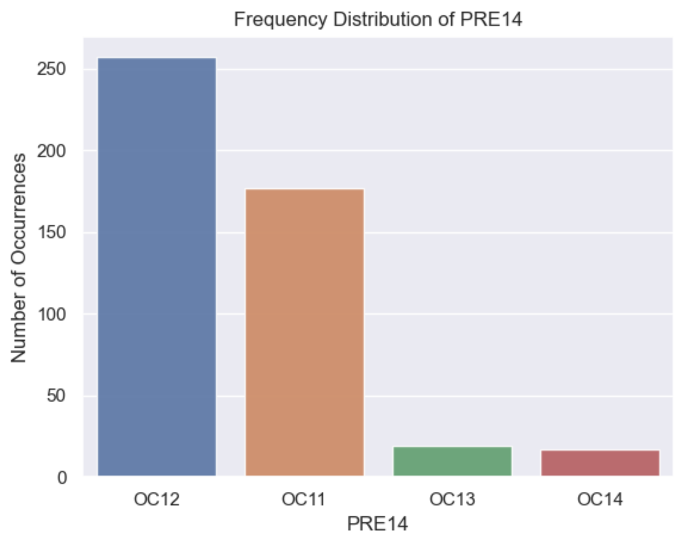
PRE6

This is the frequency distribution of PRE6. According to the graph, PRZ1 appears 313 times, PRZ0 appears 130 times, PRZ2 appears 27 times in the dataset.



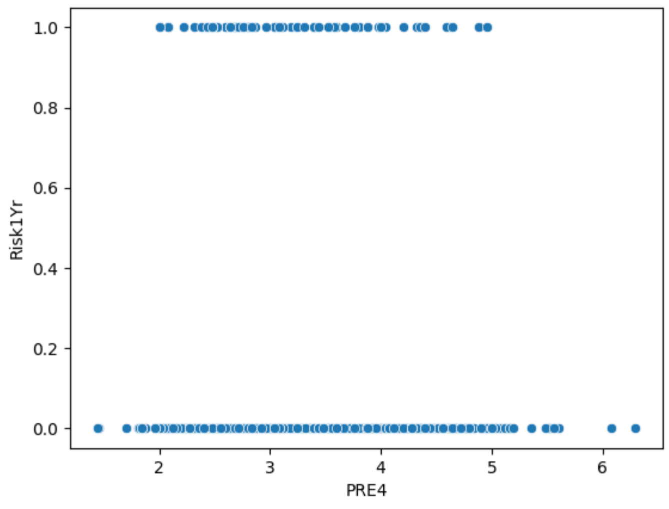
PRE14

This is the frequency distribution of PRE14. According to the graph, OC12 appears 257 times, OC11 appears 117 times, OC13 appears 19 times, OC14 appears 17 times in the dataset.



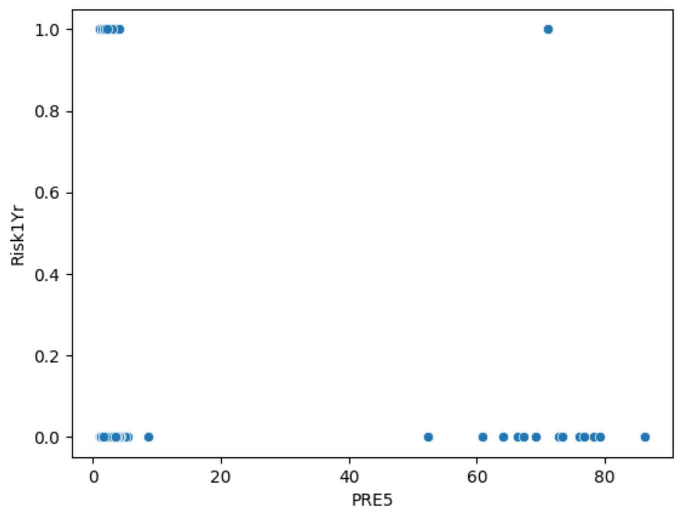
PRE4

This is the scatter plot of the numerical feature PRE4. The graph shows which class (0 or 1) PRE4 is in according to the values it takes. According to pandas profiling library’s report, PRE4 has 134 distinct values. By looking at the graph, we can say that PRE4 values smaller than 2 and PRE4 values bigger than 5 are mostly belongs to 0 class or F class.



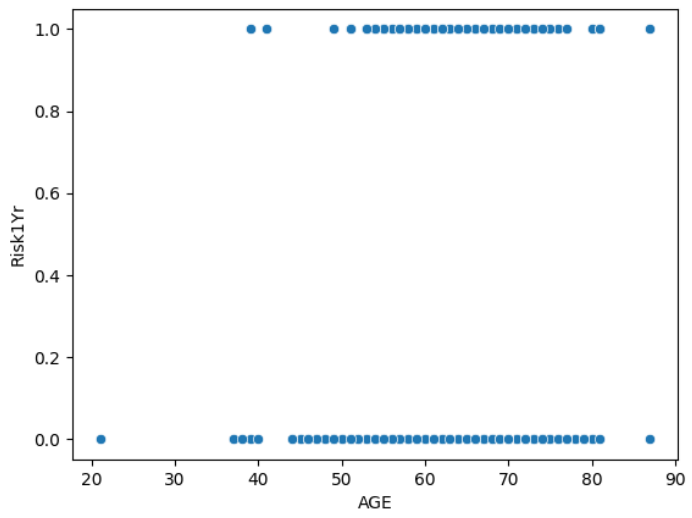
PRE5

This is the scatter plot of the numerical feature PRE5. The graph shows which class (0 or 1) PRE5 is in according to the values it takes. According to pandas profiling library's report, PRE5 has 136 distinct values. By looking at the graph, we can see that the PRE5 values greater than 50 mostly belong to the 0 or False class.

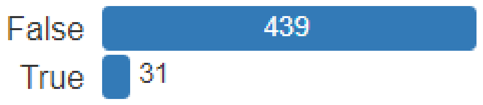


AGE

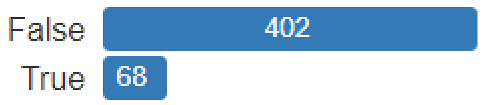
This is the scatter plot of the numerical feature AGE. The graph shows which class (0 or 1) AGE is in according to the values it takes. According to pandas profiling library's report, AGE has 45 distinct values. Although there are exceptions, we see that those younger than 50 generally belong to the 0 class. After the age of 50, it can belong to both classes.



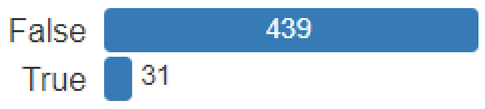
PRE7



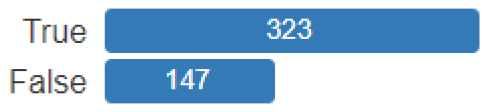
PRE8



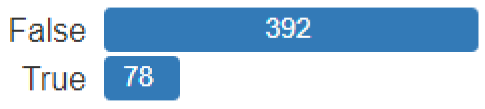
PRE9



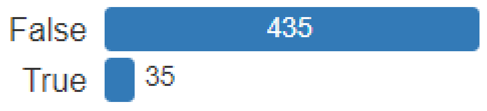
PRE10



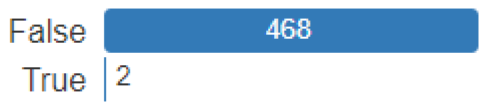
PRE11



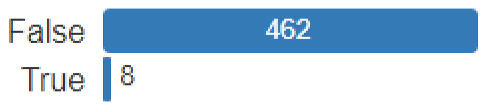
PRE17



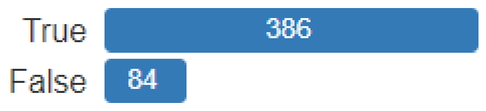
PRE19



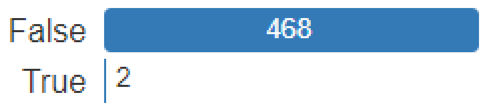
PRE25



PRE30

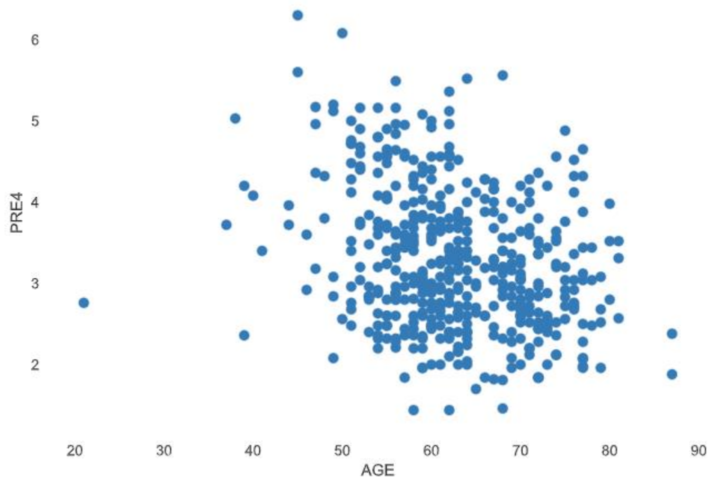


PRE32

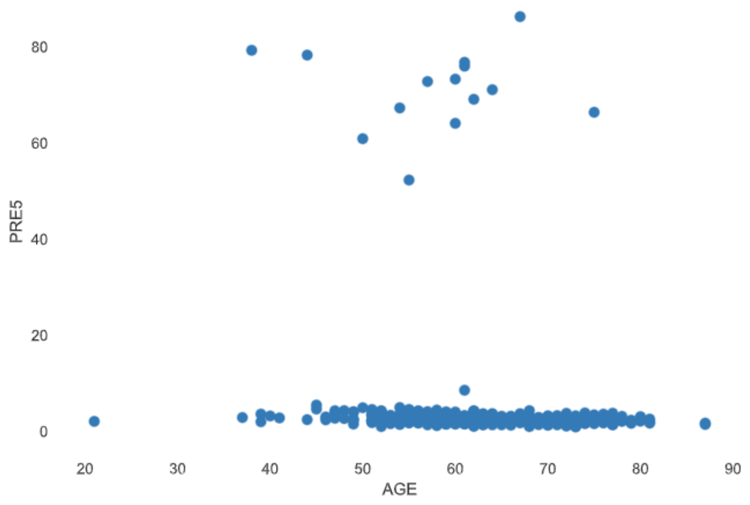


The charts above show the value distributions of Boolean features. We examined the graphs, value distributions and the results according to this distribution. Accordingly, when we evaluated our features one by one, we saw that they were not very distinctive or effective in determining the class. However, when we evaluated these features together, we saw that they became distinctive. Also, we saw interactions between PRE4, PRE5 and AGE features. So, there is no feature in our dataset that we think is redundant. Some interactions in the dataset are as follows:

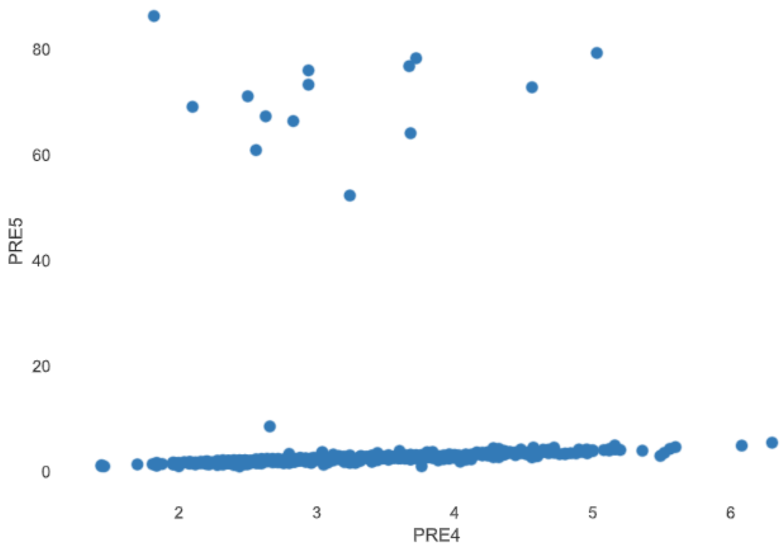
AGE – PRE4



AGE – PRE5



PRE4 – PRE5



3. Data Processing

In this section, first we handled categorical features, then normalized the data for some of the attributes we need. After scaling the data, we split the data to create train and test set. We only have 3 numeric features in our dataset, the rest is categorical feature. So, before training the data we need to handle these categorical features. To do this, we used sklearn.preprocessing LabelEncoder library. Thanks to this, we converted categorical features to numeric features. The before and after values of categorical features are as follows:

DGN Before: ['DGN2' 'DGN3' 'DGN4' 'DGN8' 'DGN5' 'DGN6' 'DGN1']

DGN After: [1 2 3 6 4 5 0]

PRE6 Before: ['PRZ1' 'PRZ0' 'PRZ2']

PRE6 After: [1 0 2]

PRE7 Before: ['F' 'T']

PRE7 After: [0 1]

PRE8 Before: ['F' 'T']

PRE8 After: [0 1]

PRE9 Before: ['F' 'T']

PRE9 After: [0 1]

PRE10 Before: ['T' 'F']

PRE10 After: [1 0]

PRE11 Before: ['T' 'F']

PRE11 After: [1 0]

PRE14 Before: ['OC14' 'OC12' 'OC11' 'OC13']

PRE14 After: [3 1 0 2]

PRE17 Before: ['F' 'T']

PRE17 After: [0 1]

PRE19 Before: ['F' 'T']

PRE19 After: [0 1]

PRE25 Before: ['F' 'T']

PRE25 After: [0 1]

PRE30 Before: ['T' 'F']

PRE30 After: [1 0]

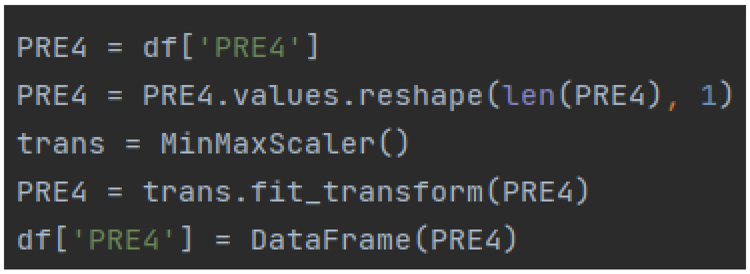
PRE32 Before: ['F' 'T']

PRE32 After: [0 1]

Risk1Yr Before: ['F' 'T']

Risk1Yr After: [0 1]

After handing the categorical features, we normalized the data of some features. We chose the PRE4 and PRE5 features to normalize, because these features are the only ones that have a large range compared to the others. To do this, we used sklearn.preprocessing MinMaxScaler library and we apply following for PRE4 and PRE5:



```
PRE4 = df['PRE4']
PRE4 = PRE4.values.reshape(len(PRE4), 1)
trans = MinMaxScaler()
PRE4 = trans.fit_transform(PRE4)
df['PRE4'] = DataFrame(PRE4)
```

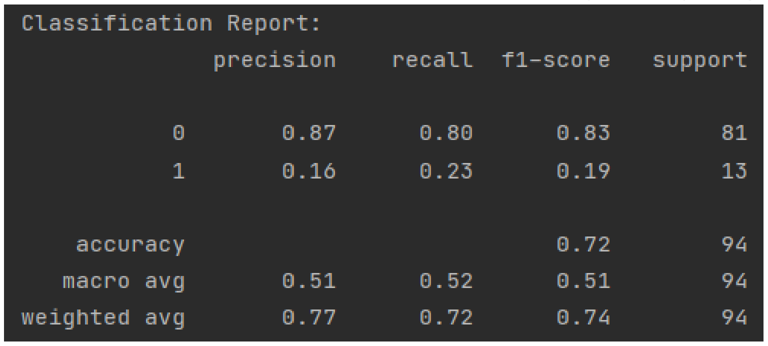
After scaling the data, we created train and test sets by splitting the data. To do this, we used sklearn.model_selection train_test_split library. We removed the Risk1Yr column from the X axis, and for the Y axis we just take the Risk1Yr column. Then we created X_train, X_test, y_train and y_test. We created test set with test_size=0.20 by default. So, we have 94 instances in the test set.

4. Model Selection and Training

In this section, we chose different models that we thought fit the classification problem. These models are decision tree classifier, logistic regression, random forest classifier and SVM classifier. To implement these models, we used sklearn.tree DecisionTreeClassifier, sklearn.linear_model LogisticRegression, sklearn.ensemble RandomForestClassifier and sklearn.svm SVC libraries. We trained models on the training set and tested on the test set, then printed the accuracies separately. Also, we used K-fold cross-validation with k = 10 and calculated RMSE scores, mean of the RMSE scores and standard deviation. According to these results, we selected the best model for our dataset.

Decision Tree Results

We used sklearn.metrics classification_library library to get this report. When we used decision



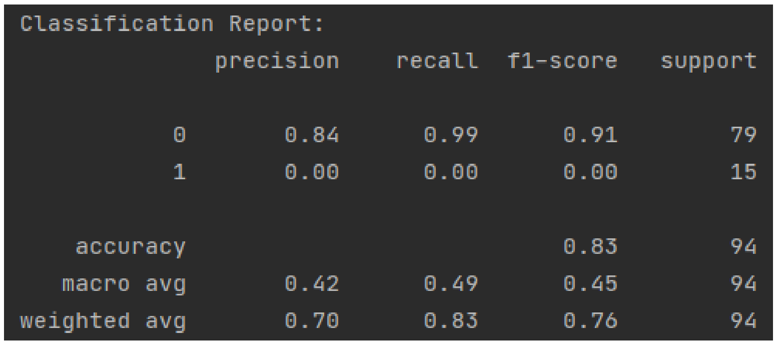
```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.80      0.83        81
           1       0.16      0.23      0.19        13

    accuracy                           0.72        94
   macro avg       0.51      0.52      0.51        94
weighted avg       0.77      0.72      0.74        94
```

tree classifier, we usually get the lowest accuracy. Accuracy score of the decision tree classifier mostly ranged from 0.70 to 0.80. Also, we used K-fold cross validation by importing sklearn.model_selecting cross_val_score library. It randomly, splits the training set into 10 distinct subsets called folds (k = 10), then it trains and evaluates the decision tree model 10 times, picking a different fold for evaluation every time and training on the other 9 folds. The result contains 10 evaluation scores and we calculated mean of these scores. We also calculated the standard deviation. The results as follows:

Mean RMSE Score = 0.4633

Standard Deviation = 0.0290

Logistic Regression Results



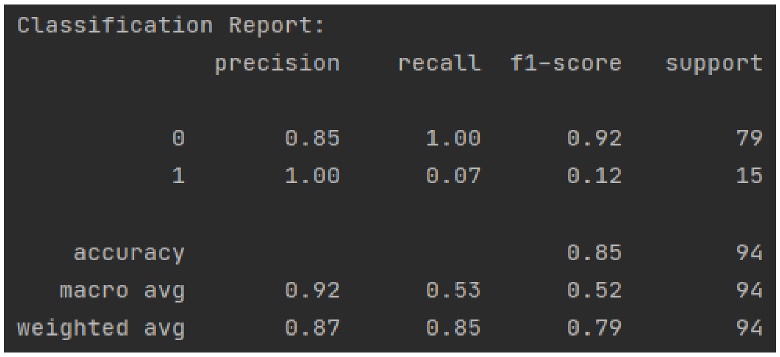
```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.99      0.91        79
           1       0.00      0.00      0.00        15

    accuracy                           0.83        94
   macro avg       0.42      0.49      0.45        94
weighted avg       0.70      0.83      0.76        94
```

When we used logistic regression, we usually get the second lowest accuracy. Accuracy score of the logistic regression mostly ranged from 0.80 to 0.85. Also, we used K-fold cross validation and calculated rmse score and standard deviation.

Mean RMSE Score = 0.3948

Standard Deviation = 0.0293

Random Forest Classifier Results



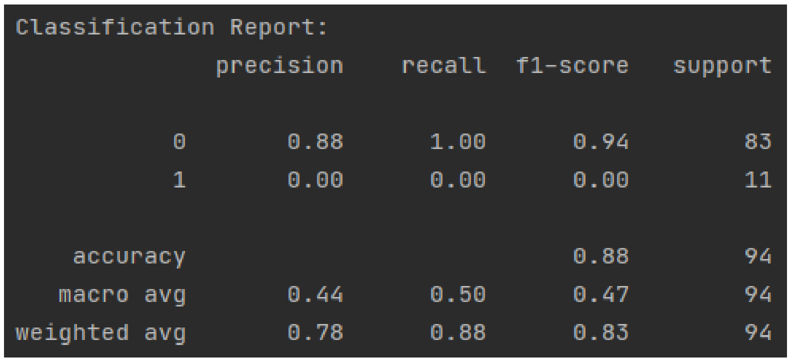
```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        79
           1       1.00      0.07      0.12        15

    accuracy                           0.85        94
   macro avg       0.92      0.53      0.52        94
weighted avg       0.87      0.85      0.79        94
```

When we used random forest classifier, we usually get the second highest accuracy. Accuracy score of the random forest classifier mostly larger than 0.85. Also, we used K-fold cross validation and calculated rmse score and standard deviation.

Mean RMSE Score = 0.3854

Standard Deviation = 0.0145

SVM Classifier Results



```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        83
           1       0.00      0.00      0.00        11

    accuracy                           0.88        94
   macro avg       0.44      0.50      0.47        94
weighted avg       0.78      0.88      0.83        94
```

Although SVM classifier and random forest classifier accuracy results are close to each other, SVM classifier usually gives higher accuracy. In addition, the standard deviation result of the SVM classifier is lower.

Mean RMSE Score = 0.3959

Standard Deviation = 0.0097

Although the RMSE score is slightly higher than the random forest classifier and logistic regression models, its accuracy is higher, and its standard deviation is lower than the others. For this reason, we determined the SVM classifier as the best model for our dataset.

5. Fine-tune Your Model

In this section, we fine-tuned our selected model. That is, we selected the best hyperparameters for the model. To do this, we used sklearn.model_selecting GridSearchCV library. We wrote the hyperparameters that we want to try and thanks to this library we evaluated possible combinations of hyperparameter values. Then, we found the best hyperparameters for the model. Also, we re-ran predictions with these best hyperparameters and got the classification report.

In SVM classifier we have 3 hyperparameters, these are C, gamma and kernel. SVM creates a decision boundary which makes the distinction between classes and tries to separate all the positive and negative classes. When determining the decision boundary, a soft margin SVM tries to solve an optimization problem with the following goals:

- Increase the distance of decision boundary to classes.
- Maximize the number of points that are correctly classified in the training set.

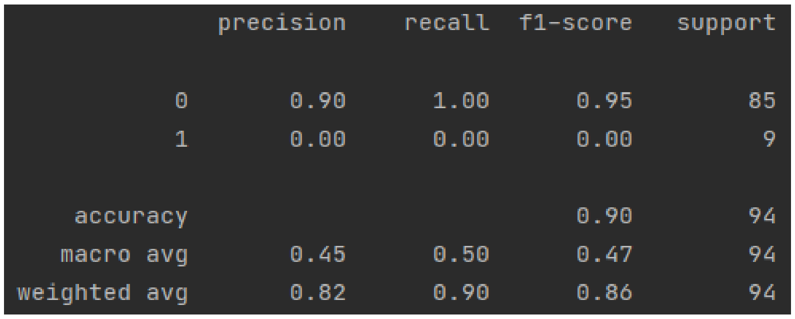
Hyperparameter C allows to define the tradeoff between these two goals. A smaller value of C leads to a decision boundary with a large margin but more margin violations. A larger value of C leads to a decision boundary with a smaller margin and increasing the value of C increases the variance and lowers the bias of the model. Gamma is a hyperparameter used with non-linear SVM. One of the most used non-linear kernels is the radial basis function (RBF). Gamma parameter of RBF controls the distance of the influence of a single training point. Low values of gamma indicate a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other to be considered in the same class.

According to the results, best hyperparameters after tuning are as follows:

C = 0.1

Gamma = 1

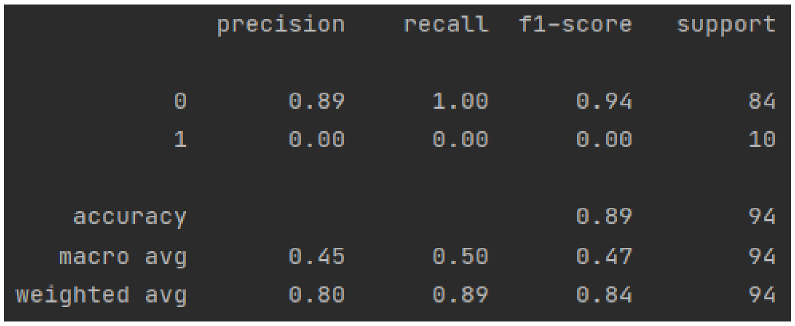
Kernel = rbf



```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95        85
           1       0.00      0.00      0.00         9

    accuracy                           0.90        94
   macro avg       0.45      0.50      0.47        94
weighted avg       0.82      0.90      0.86        94
```

6. Testing

In this section, we tested our model on the test set. We created classification report which includes accuracy, precision, recall and F-score.



```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        84
           1       0.00      0.00      0.00        10

    accuracy                           0.89        94
   macro avg       0.45      0.50      0.47        94
weighted avg       0.80      0.89      0.84        94
```

True Negative (TN): The actual value is False, and the model predicted as also False.

False Positive (FP): The actual value is False, but the model predicted as True.

False Negative (FN): The actual value is True, but the model predicted as False.

True Positive (TP): The actual value is True, and the model predicted as also True.

Precision = $\frac{TP}{TP+FP}$

Recall = $\frac{TP}{TP+FN}$

F1-score = $2.\left(\frac{Precision.Recall}{Precision + Recall}\right)$

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

According to the results, we got 0.89 for precision, 1 for recall, 0.94 for F1-score and 0.89 for accuracy. Although our accuracy value is slightly lower than the previous one, it is higher than before fine-tuning.

7. Summary

In this project, we learned how to analyze and process the raw data, and then we learned the steps we need to follow to choose the best model for this dataset. To achieve our goal, we first analyzed our dataset and understood what we needed to do. All features in our dataset, except for only 3 numerical features, are categorical. Therefore, we first need to handle these features to use them in the model selection process. We converted the values of these features to numeric values using LabelEncoder. Then we need to normalize or scale some numerical features to get more accurate and better results. In order to normalize it, we chose PRE4 and PRE5 features, which have a wider range in our dataset. Then, we split the data into train and tests set to use them in model training and testing. After creating test and train set, we implemented decision tree classifier, logistic regression model, random forest classifier and SVM classifier. Since our problem is a classification problem, we chose these models and thought that these models could be suitable for the data set. We trained and tested these models and using K-fold cross-validation we selected the one model with the best result. According to the results, the decision tree classifier was mostly the model that gave the lowest accuracy. The other three models gave similar accuracy, so we looked at the RMSE scores and saw that the logistic regression model had a higher RMSE score than the others. For this reason, we eliminated the decision tree classifier and logistic regression models. When we look at the last two models, we saw that the SVM classifier gave a slightly higher accuracy, and we also saw that the standard deviation was lower. Thus, we eliminated the random forest classifier and selected SVM classifier as the best model. Then we fine-tuned this model and found the best hyperparameters for the model. Finally, we tested our final model on the test set and calculated accuracy, precision, recall and f1-score results.