Mariam Kili Bechir

Machine Learning Project Documentation

Model Refinement

1. Overview

In the pursuit of accurate chronic disease prediction using machine learning, the model refinement phase is pivotal, serving as the compass that navigates us toward optimal performance. Given the classification nature of disease prediction, where outcomes are binary (0 for unaffected, 1 for affected), and acknowledging the diverse characteristics of three distinct datasets, carefully selecting classification algorithms is crucial.

For heart disease dataset, the best model was found with Logistic Regression, which is a reliable choice for its simplicity, computational efficiency, and resistance to overfitting, especially with a limited feature set. Its interpretability, direct modeling of output probability, and robustness make it a natural starting point.

For diabetes disease and Parkinson disease datasets, the algorithms used are Logistic Regression, SVM, RandomForest, KNN and XGboost. SVM's effectiveness in high-dimensional spaces and KNN's flexibility in capturing complex decision boundaries offer intriguing alternatives. RandomForest's robustness and feature importance scores, and XGBoost's gradient boosting prowess bring added layers of sophistication.

2. Model Evaluation

For heart disease dataset, the best model was found with Logistic Regression, the score obtained is 86% on the training data and 89% on the test dataset. 20% of our dataset is considered for evaluating and testing the model. The training score with SVM is 86% on the training data and 86% on the test data.

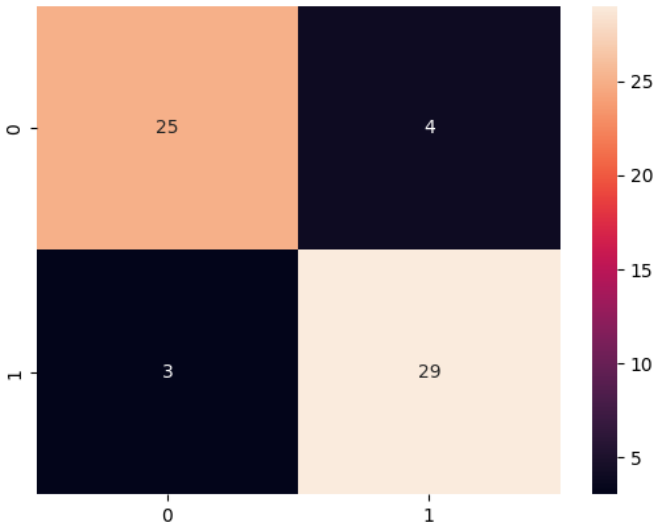


Figure 1-confusion_matrix with Logistic Regression for heart dataset

Figure 2-ROC Curve for Logistic Regression with heart dataset

For diabetes disease dataset, the algorithms used are Logistic Regression, SVM, RandomForest and XGboost. The training score with Logistic Regression is 79% and the test score is 71%. With SVM, the accuracy on training dataset is 79% and 72% on test data.The training score using RandomForest is 100% and the test score with with the same algorithm was 77%. The training score XGBoost is 100% and the test score with XGBoost is 75%.

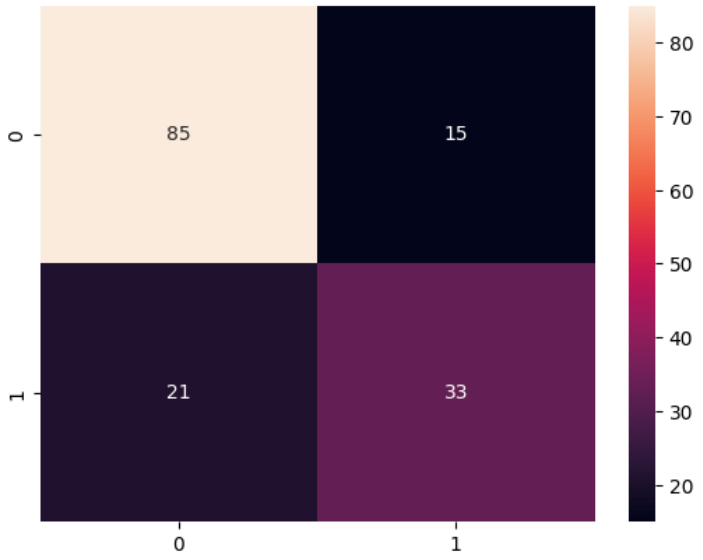


Figure 3-confusion matrix with RandomForest for diabetes dataset

Figure 4-ROC Curve with RandomForest for diabetes dataset

For Parkinson disease dataset, the Machine Learning techniques used are: Logistic Regression, SVM, KNN, RandomForest and XGBoost. All these algorithms gave a good result but the best one obtained is with KNN. The training score KNN is 95% and the test score is also 95%. The training score RandomForest and XGBoost is 100% and the test score is 95% for both the models.

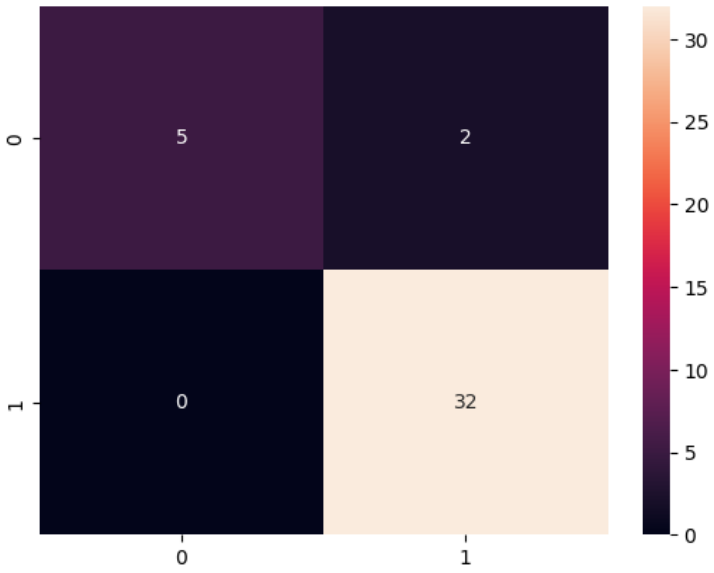


Figure 5-confusion matrix with KNN for Parkinson dataset

Figure 6-ROC Curve with KNN for Parkinson dataset

3. Refinement Techniques

Hyperparameter Tuning, specifically GridSearch with Cross-Validation (CV) equal to 5, was employed for heart disease, refining Logistic Regression, RandomForest, and SVM. Similar techniques were used for diabetes and Parkinson's datasets, optimizing SVM, KNN, RandomForest, and XGboost.

4. Hyperparameter Tuning and Cross-Validation

For heart disease dataset, the best model was found with Logistic Regression, the score obtained is 86% on the training data and 89% on the test dataset. These are the results obtained after using Hyper Parameter Tuning to optimize our model. The hyperparameter tuning technique used is GridSearch with Cross-Validation equal to 5.

Cross-validation can be used in conjunction with GridSearch to fine-tune hyperparameters in a machine learning model. Cross-validation helps to robustly estimate the performance of a model by splitting the data into multiple subsets, training the model on different combinations of these subsets, and evaluating its performance on the remaining data. With this technique, we combined different algorithms together with their different hyperparameters and the best algorithms was Logistic Regression, RandomForest and Support Vector Machine (SVM).

	model	**Best score**	**best_parameters**
0	logistic_regression	0.885245	C= 0.23, max_iter= 100
1	random_forest	0.8852459	max_depth=2, max_leaf_nodes= 6
3	SVM	0.86885245	C= 100, gamma= 1, kernel= 'linear'

4	xg_boost	0.797619	max_depth=9, max_leaf_nodes=3
5	KNN	0.702381	metric= 'manhattan', 'n_neighbors': 7

Table 1-Hyperparamter tuning result of heart disease data

Classificatio_report for LogistciRegression

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61

Table 2- Classification report for Logistic Regression on heart data

For diabetes disease dataset, the algorithms used are Logistic Regression, SVM, KNN, Random Forest and XGboost. XGBoost gave the best score after Hyper Parameter Tuning. The score obtained by this algorithm is 98% on the training dataset and 79% on the test dataset. Hyper parameter Tuning improved the accuracy of our model on training and testing data. The training score with RandomForest gave us 99% and the test score with the same algorithm gave 78% as a result.

	model	**best_score**	**best_parameters**
0	svm	0.780168	{'C': 10, 'kernel': 'linear'}
1	knn	0.762255	{'n_neighbors': 10}
2	random_forest	0.780155	{'max_depth': 11, 'max_features': 'log2', 'n_es...
3	xg_boost	0.792207	{'max_depth': 10, 'n_estimators'=15)
4	logistic_regression	0.781781	{'C': 5, 'n_jobs': -1}

Table 3-hyperparamter tuning result for diabetes dataset

Classificatio_report for XGBoost

	precision	recall	f1-score	support
0	0.81	0.89	0.85	100
1	0.75	0.61	0.67	54
accuracy			0.79	154
macro avg	0.78	0.75	0.76	154
weighted avg	0.79	0.79	0.79	154

Table 4-classification report for XGboost on diabetes data

For Parkinson disease dataset, the Machine Learning techniques used are: Logistic Regression, SVM, KNN, RandomForest and XGBoost. All these algorithms gave a good result but the best one obtained is with KNN. The scores obtained using KNN are 92 % on the training dataset and 90% on the test data. The second-best algorithm for this data is SVM. The training score with SVM is: 85% and the test score with SVM is 87%. GridSearch hyper Parameter tuning with cross-validation technique was used to perform the result and get this final score.

	model	**best_score**	**best_parameters**
0	svm	0.910081	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}
1	knn	0.897581	{'metric': 'manhattan', 'n_neighbors': 11
2	random_forest	0.878024	{'max_depth': 10, 'max_leaf_nodes': 9
3	xg_boost	0.884476	{'max_depth': 6, 'max_leaf_nodes': 3
4	logistic_regression	0.833266	{'C': 0.012742749857031334, 'max_iter': 100

Table 5-hyperparamter tuning result for parkinson dataset

Looking at the hyperparameter result, the best algorithm that we got is SVM, but we tried KNN without any parameter and we got 95% as accuracy. So we conclude that KNN is the best model for this data.

Classificatio_report for KNN

	precision	recall	f1-score	support
0	1.00	0.71	0.83	7
1	0.94	1.00	0.97	32
accuracy			0.95	39
macro avg	0.97	0.86	0.90	39
weighted avg	0.95	0.95	0.95	39

Table 6-classification report fro KNN on Parkinson data

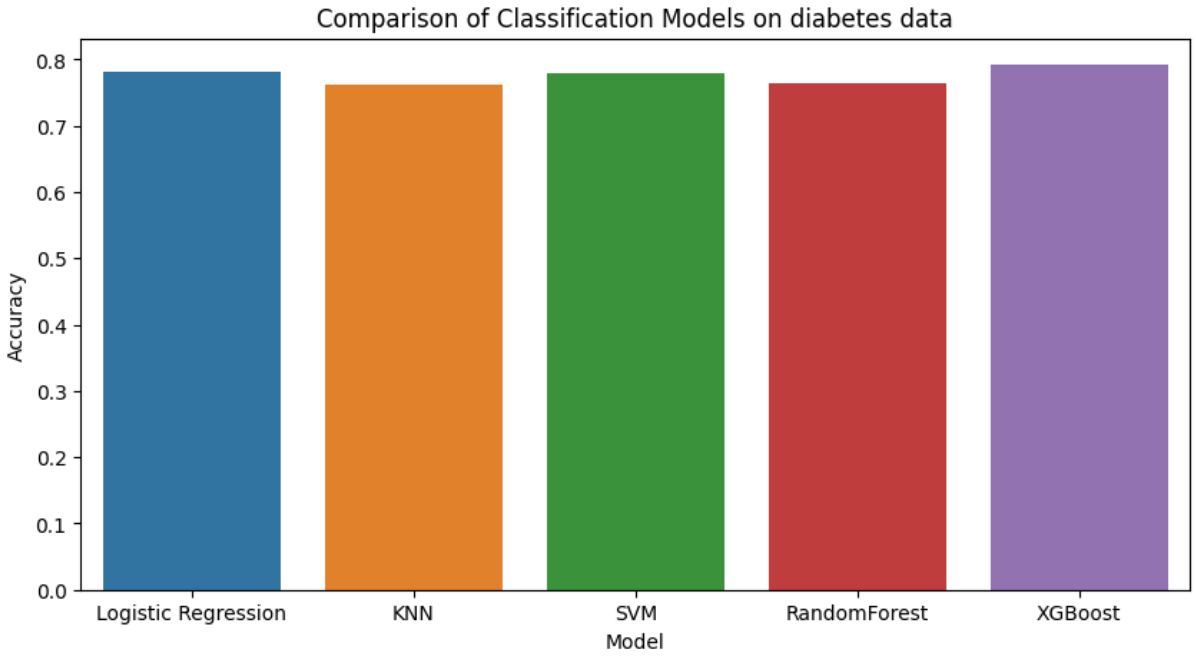
6. Feature Selection

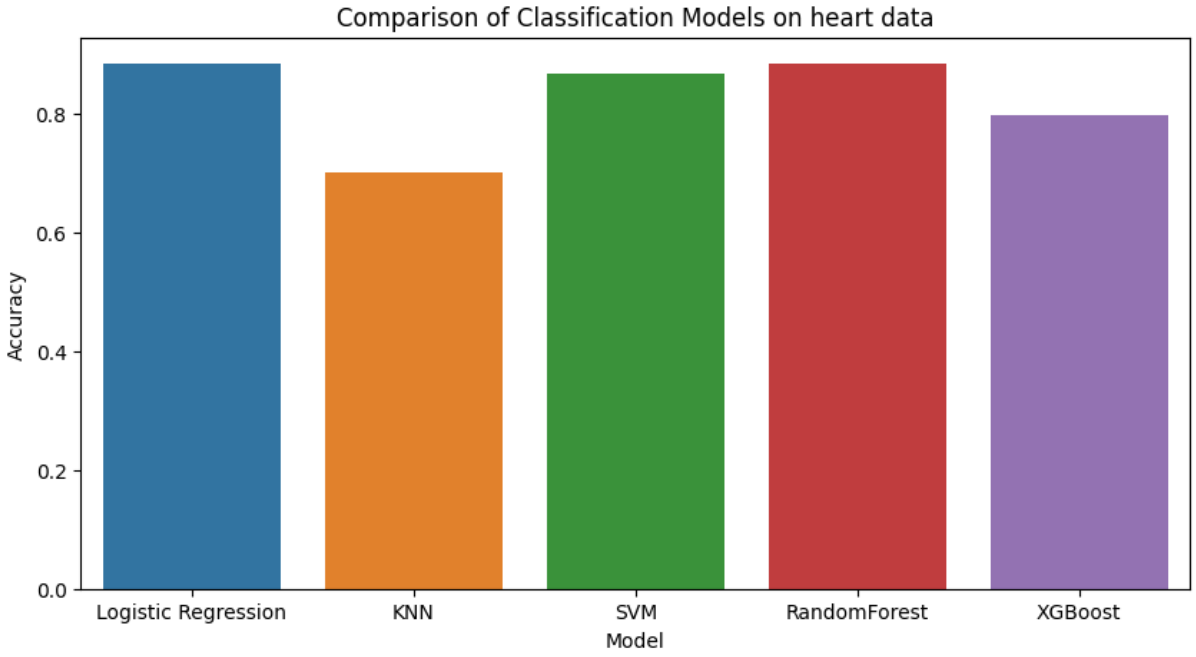
In this project, three different datasets are used. Each of these datasets contains numerical values, except Parkinson dataset which contains, ‘name’ feature which is a categorical feature. We don’t need this categorical feature to train the data, so we dropped this feature. While training the models, we also used Random Forest and XGBoost which are Tree-based algorithms that naturally provide feature importance scores.

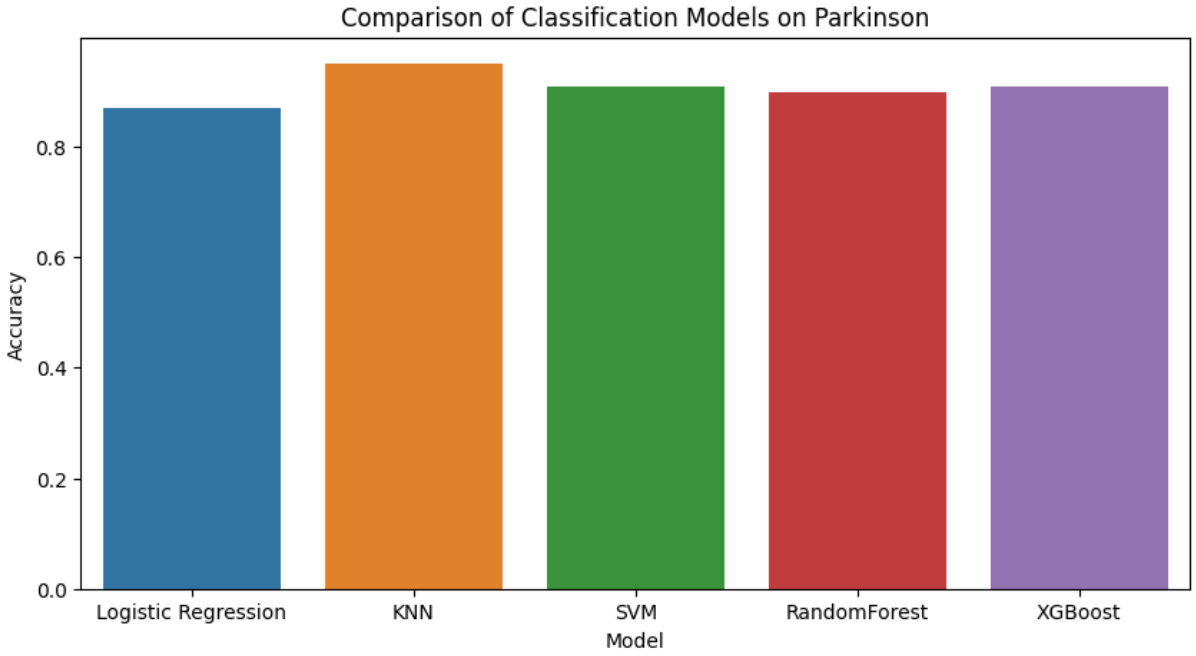
For diabetes dataset, we used feature_importances_ of XGBoost to see the the most important features and select them and then analyse the score obtained after retraining the model. We obtained 8 features selected but the accuracy was 75% on the test dataset. So, we just considered the initial features of our dataset and we considered the high accuracy obtained without feature selection technique.

For heart disease dataset, we used Recursive Feature Elimination with Cross-Validation (RFECV) feature selection technique. RFECV is a feature selection technique that combines the concepts of recursive feature elimination (RFE) and cross-validation. The goal of RFECV is to find the optimal number of features by recursively removing the least important features and evaluating the model performance using cross-validation at each step. 10 features were selected and the score on test data is 89% with Logistic Regression.

No feature selection method is applied for Parkinson dataset because the final accuracy obtained is quite enough: 95%.







Test Submission

1. Overview

Before getting into the deployment of the project, we tested our three models using simple data by respecting the number of features of each dataset. We obtained a positive or negative response according to the values given by the user as inputs. We then build the web app using Streamlit python library and we deployed our models using this library in the streamlit cloud service.

2. Data Preparation for Testing

The data that we took to test our models should just be an array of continuous values. The initial data used for training also was defined as numpy array of continuous values.

Testing Model

```
input_data = (85,0,1,85,80,0,0,10,0,3.6,1,5,5)
#input_data=(63, 1, 3, 145, 233, 1, 0, 150, 0, 2.3, 0, 0, 1)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
    print('The Person is not affected by Heart Disease')
else:
    print('The Person has Heart Disease')
```

```
[0]
The Person is not affected by Heart Disease
```

Model testing for diabetes disease

```
#input_data = (5,166,72,19,175,25.8,0.587,51)\n",
input_data=(1, 89, 66, 23, 94, 28.1, 0.167, 21)
#input_data = (5,166,72,19,175,25.8,0.587,51)\n",

# changing the input_data to numpy array\n",
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance\n",
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = loaded_model.predict(input_data_reshaped)
print(prediction)
if (prediction[0] == 0):
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

```
[1]
The person is diabetic
```

Model Testing for Parkinson disease ¶

```
#input_data = (10,5)
input_data=(119, 157, 75, 0.05, 0.0005, 0.003, 0.007, 0.002, 0.04, 0.05, 0.007, 0.02, 0.006,22,0.5,0.8,1,0.5,2.310,0.284, 1, 2)
#119.992——157.302 74.997—0.00784 0.00007 0.00370 0.00554 0.01109 0.04374 0.06545 0.02211 21.033—1——0.414783——0.815285——

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model2.predict(input_data_reshaped)
print(prediction)

if (prediction[0] == 0):
    print('The person is affected by Parkinson')
else:
    print('The person is not affected by parkinson')
```

```
[0]
The person is affected by Parkinson
```

3. Model Application

When we selected the best model for each of our dataset, the model was then saved using 'joblib' python library. Joblib is a python package used to save and load machine learning models. Joblib is especially useful for machine learning models because it allows you to save the state of your computation and resume your work later or on a different machine. Since data is stored as byte strings rather than objects, it may be stored quickly and easily in a smaller amount of space than traditional pickling. Joblib module in Python is especially used to execute tasks parallelly using Pipelines rather than executing them sequentially one after another. Joblib module lets the user use the full potential of their devices by utilizing all the cores present in their device to make the process as fast as possible.

```
import joblib
filename = 'diabetes_model.joblib'
joblib.dump(model2, open(filename, 'wb'))
```

```
# loading the saved model\n",
loaded_model = joblib.load(open('diabetes_model.joblib', 'rb'))
```

Figure 7-save and load Random forest model

```
import joblib
filename = 'heart_model.joblib'
joblib.dump(model1, open(filename, 'wb'))
```

```
# loading the saved model
loaded_model = joblib.load(open('heart_model.joblib', 'rb'))
```

Figure 8-save and load Logistic Regression model

```
import joblib
filename = 'parkinson_model.joblib'
joblib.dump(model2, open(filename, 'wb'))
```

```
# loading the saved model\n",
loaded_model = joblib.load(open('parkinson_model.joblib', 'rb'))
```

Figure 9-save and load KNN model

After saving the model, in order to use it again and be able to make prediction on any machine using external data, we loaded the models again and test them with some data to see the result of the trained models.

4. Test Metrics

The evaluation metrics used to test our machine learning models are, acurracy_score of scikit-learn, confusion matrix, classification_reports, AUC-ROC curve. The prediction was performed on the test data which represent 20% of each dataset. After the predictions, accuracy-score metrics was used on the test data and the training data to see the performance of the models on both, training and testing data.

For the XGBoost model trained on diabetes dataset, the accuracy score on test data was 78% and 98% on the training dataset. Here is the ROC-AUC (Receiver Operating Characteristics-Are Under the curve) curve. ROC curve is the graphical representation of the effectiveness of the binary classification model. It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds. AUC stands for Area Under the Curve, and the AUC curve represents the area under the ROC curve. It measures the overall performance of the binary classification model. As both TPR and FPR range between 0 to 1, So, the area will always lie between 0 and 1, and A greater value of AUC denotes better model performance. Our main goal is to maximize this area in order to have the highest TPR and lowest FPR at the given threshold.

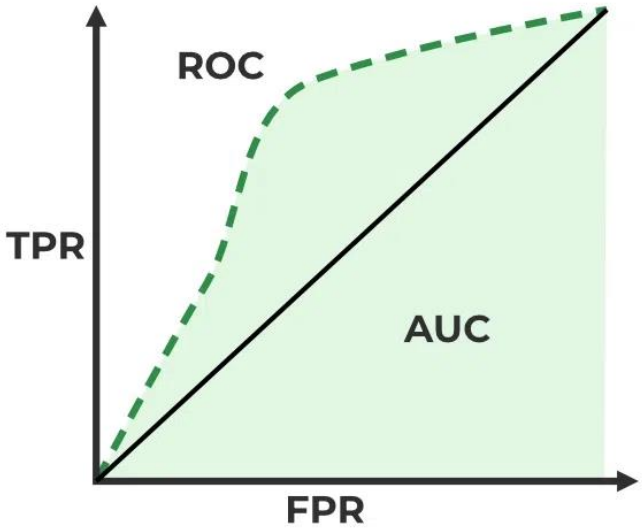


Figure 10ROC-AUC Classification Evaluation Metric

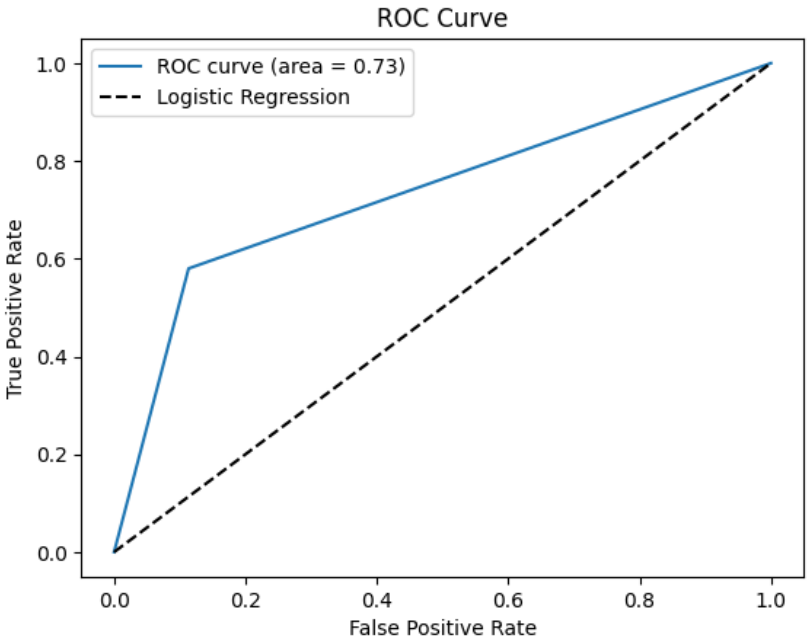


Figure 11-ROC-curve for diabetes test data

For heart disease dataset, the best model was found with Logistic Regression, the score obtained is 86% on the training data and 89% on the test dataset.

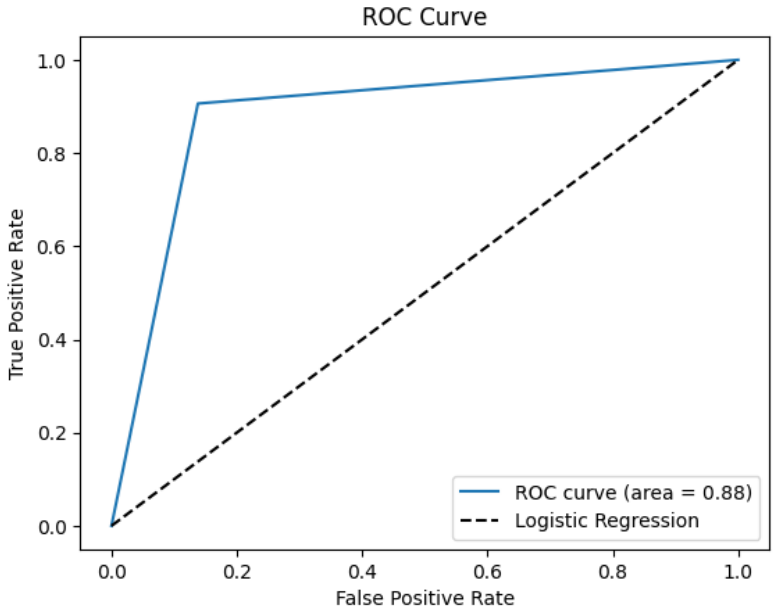


Figure 12-ROC Curve for heart disease test data

For Parkinson disease dataset, the Machine Learning techniques used are: Logistic Regression, SVM, KNN, RandomForest and XGBoost. All these algorithms gave a good result but the best one obtained is with KNN. The scores obtained using KNN are 95 % on the training dataset and 95% on the test data.

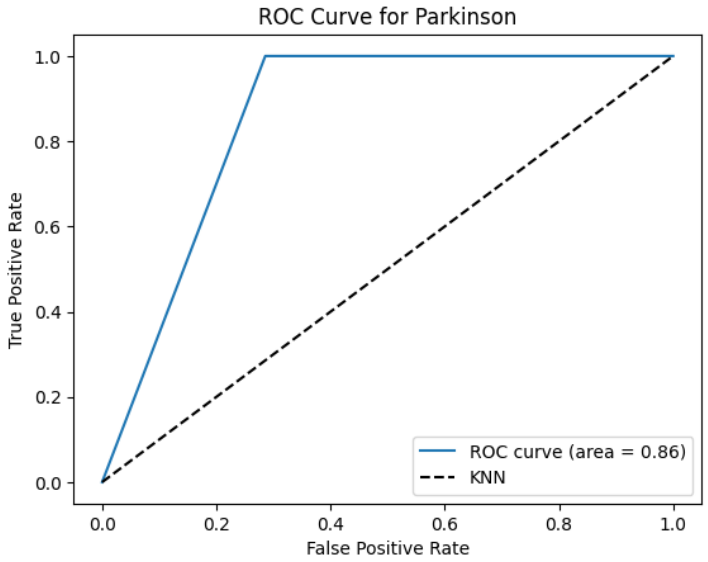


Figure 13-ROC curve for Parkinson test data

5. Model Deployment

The application built is an app that regroups the three different Chronic diseases that we want to test. After that, this application was deployed using the deployment feature of streamlit and github. Our application was connected to the github repository" https://github.com/Kili66/Capstone_DataScience_Project_SDG" that I have created for this project. Streamlit analysis this repository by checking if the models work properly and then deploy our application using the streamlit cloud: https://capstonedatascienceprojectsdg-lt7uswtnqcyahlogqcejma.streamlit.app/.

6. Code Implementation

Model chosen for diabetes dataset

```
#from xgboost import XGBClassifier\n",
#model2= XGBClassifier(max_depth=9)\n",

#model4= XGBClassifier(max_depth=10, n_estimators=15)
model4= XGBClassifier(max_depth=6, n_estimators=15)

model4.fit(X_train, y_train)
y_pred_xgb= model4.predict(X_test)
print("The training score with Xgboost is:", model4.score(X_train, y_train))
print("The test score with Xgboost is:", model4.score(X_test, y_test))
```

```
The training score with Xgboost is: 0.9726495726495726
The test score with Xgboost is: 0.782312925170068
```

Figure 14-code snippet for Xgboost model on diabetes dataset

```
model1= LogisticRegression(C= 0.23, max_iter=100)
model1.fit(X_train, y_train)
y_pred_lr= model1.predict(X_test)
```

```
print("The training score with LogisticRegression is:", model1.score(X_train, y_train))
print("The test score with LogisticRegression is:", model1.score(X_test, y_test))
```

```
The training score with LogisticRegression is: 0.859504132231405
The test score with LogisticRegression is: 0.8852459016393442
```

Figure 15-code snippet for Logistic Regression model on gerat disease dataset

```
model2= KNeighborsClassifier()
model2.fit(X_train, y_train)
y_pred_knn= model2.predict(X_test)
```

```
print("The training score with KNN is:", model2.score(X_train, y_train))
print("The tEST score with kNN is:", accuracy_score(y_test, y_pred_knn))
```

```
The training score with KNN is: 0.9487179487179487
The tEST score with kNN is: 0.9487179487179487
```

Figure 16-code snippet for KNN model on parkinson dataset

Conclusion

The model refinement phase, incorporating Hyperparameter Tuning and careful algorithm selection, significantly improved prediction accuracy. Challenges were addressed through systematic testing and feature selection methods, contributing to robust models. The final deployment ensures accessibility for users to benefit from accurate chronic disease predictions, aligning with the project's broader goals. The outcomes underscore the effectiveness of machine learning in enhancing healthcare decision-making.

References

1- https://www.analyticsvidhya.com/blog/2023/02/how-to-save-and-load-machine-learning-models-in-python-using-joblib-library/
2- https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/
3- https://www.geeksforgeeks.org/auc-roc-curve/
4- S. Vilas and A. M. S. Scholar, 'Diseases Prediction Model using Machine Learning Technique', doi: 10.32628/IJSRST.
5- D. Hamid, S. S. Ullah, J. Iqbal, S. Hussain, C. A. U. Hassan, and F. Umar, 'A Machine Learning in Binary and Multiclassification Results on Imbalanced Heart Disease Data Stream', *J Sens*, vol. 2022, 2022, doi: 10.1155/2022/8400622.
6- M. ÇOLAK, T. TÜMER SİVRİ, N. PERVAN AKMAN, A. BERKOL, and Y. EKİCİ, "Disease prognosis using machine learning algorithms based on new clinical dataset," *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, vol. 65, no. 1, pp. 52–68, Jun. 2023, doi: 10.33769/aupse.1215962.
7- R. Bharti, A. Khamparia, M. Shabaz, G. Dhiman, S. Pande, and P. Singh, "Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning," *Comput Intell Neurosci*, vol. 2021, 2021, doi: 10.1155/2021/8387680.