Capstone Project 2 ‘Early Stage Diabetes Risk Prediction’

SPRINGBOARD – DSC

By Igor Verevkin

December 2020

Problem Identification

DIABETES

- 442 million people have diabetes
- 1.6 million deaths every year
- Important to diagnose early
- What are early signs?

GOAL

- Build predictive models
- Estimate the probability that a patient is diabetic

Data Acquisition

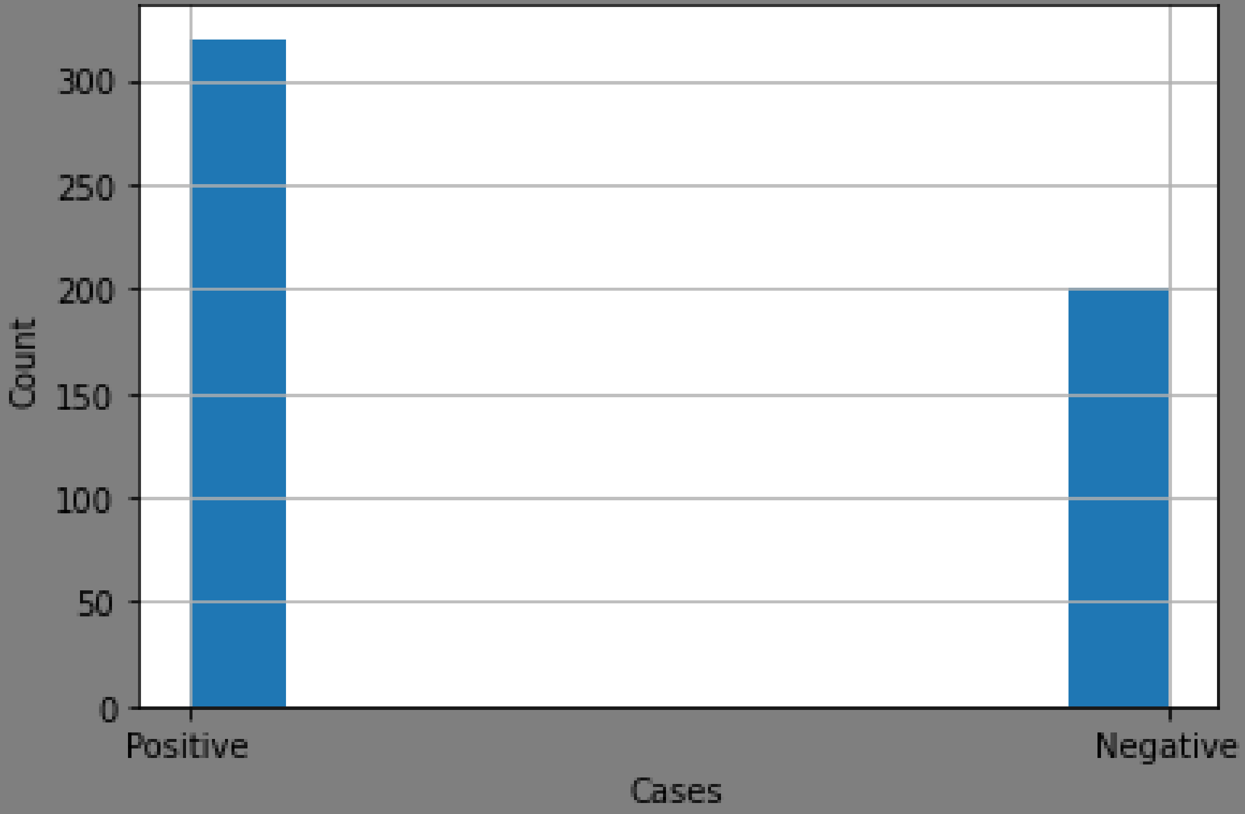
SOURCE

- Metropolitan University Sylhet, Bangladesh
- UC Irving machine learning repository

DATA WRANGLING

- Dataset
  - Observations: patients
  - Exploratory Features: patient's characteristics and symptoms
  - Target Feature: patient's class (positive (e.g. diabetic) or negative (e.g. non-diabetic))
- No missing values
- Dataset unbalanced - more positive values (320) than negative (200)

Dataset is unbalanced

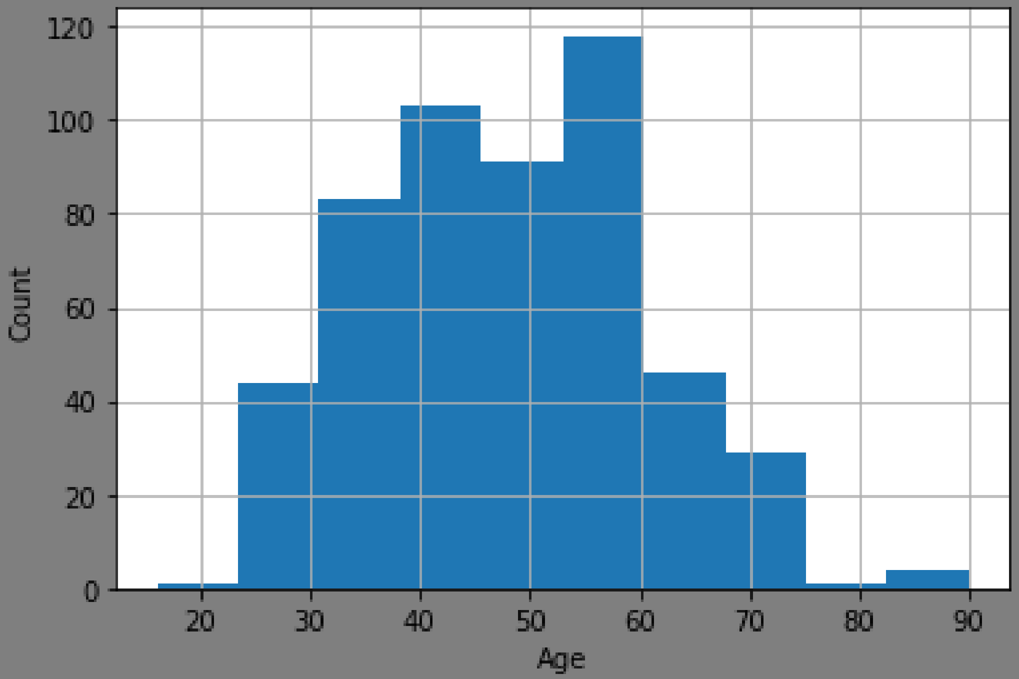


Exploratory Data Analysis

QUESTIONS:

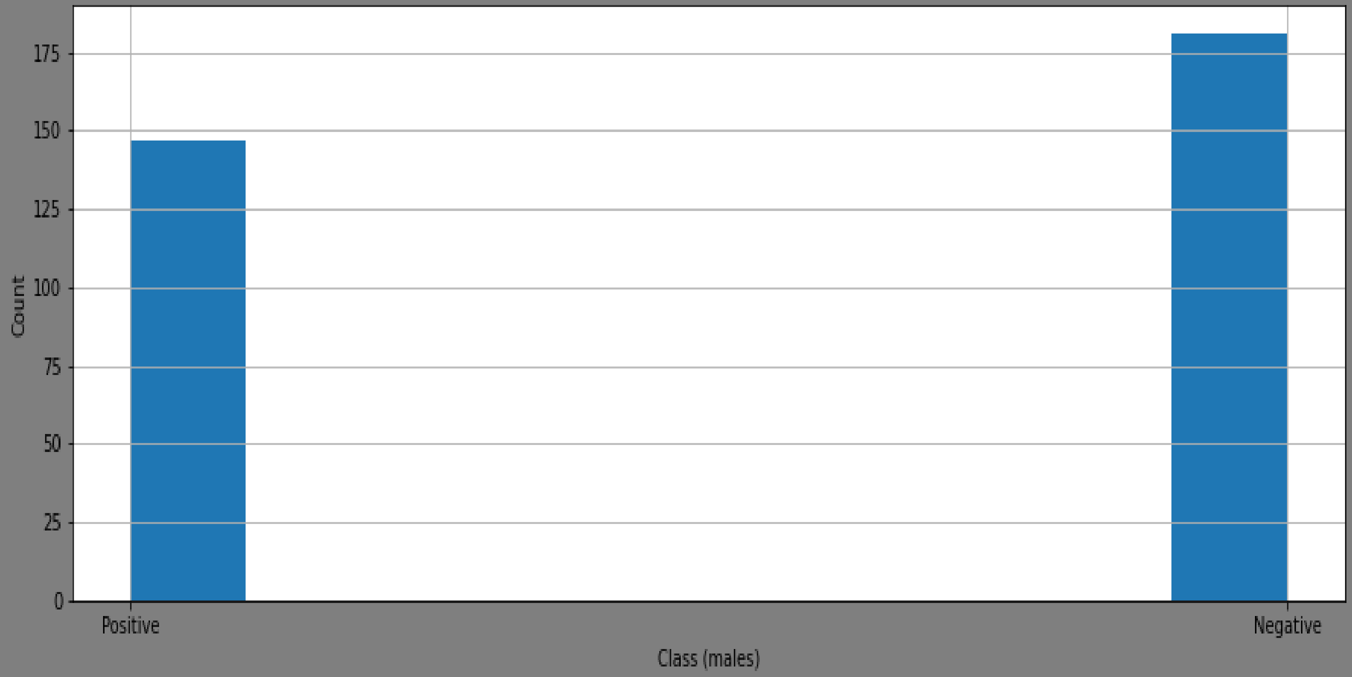
- Average, maximum and minimum age?
- Which gender has more positive cases?
- Feature-to-target correlations?

Age Distribution



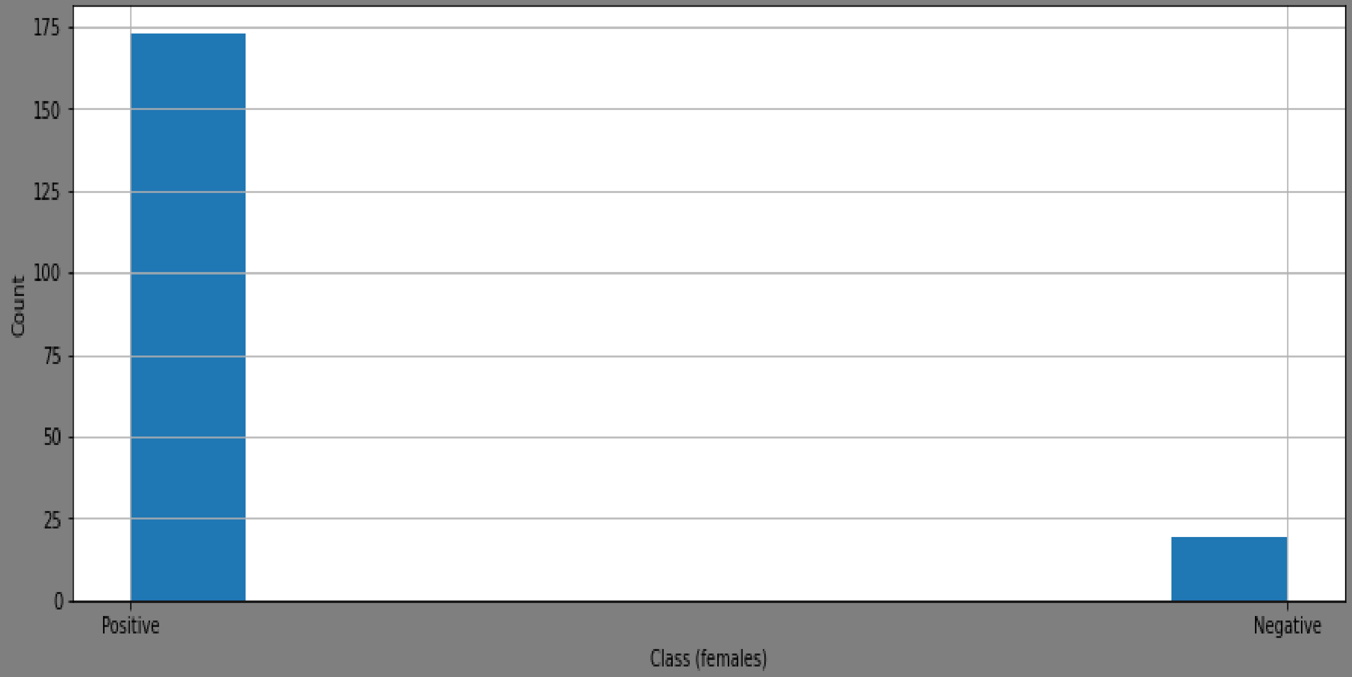
- Average age – 48 yeas old
- Minimum age – 12 years old
- Maximum age – 92 years old

Cases distribution (male)



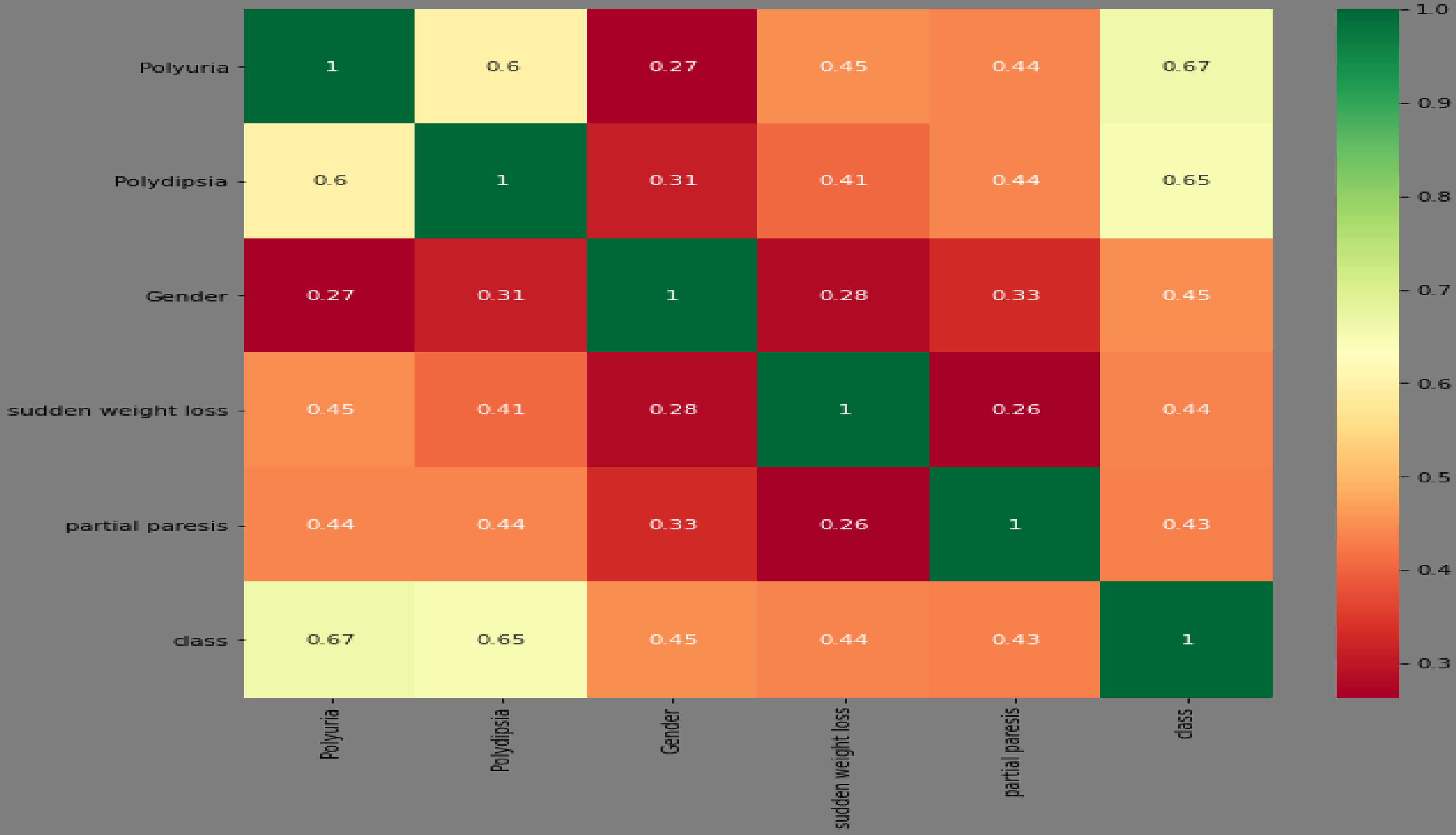
- Positive cases – 147 (44.8%)
- Negative cases – 181 (55.2%)

Cases distribution (female)



- Positive cases – 173 (90.1%)
- Negative cases – 19 (9.9%)

Feature-to-target correlation



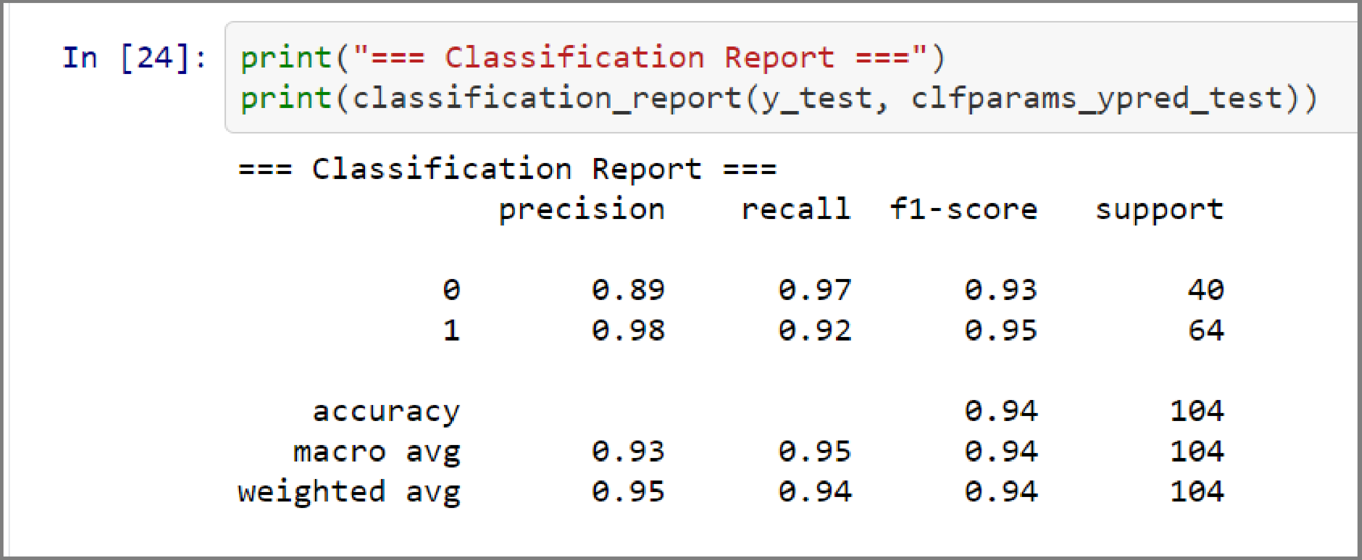
	Polyuria	Polydipsia	Gender	sudden weight loss	partial paresis	class
Polyuria	1	0.6	0.27	0.45	0.44	0.67
Polydipsia	0.6	1	0.31	0.41	0.44	0.65
Gender	0.27	0.31	1	0.28	0.33	0.45
sudden weight loss	0.45	0.41	0.28	1	0.26	0.44
partial paresis	0.44	0.44	0.33	0.26	1	0.43
class	0.67	0.65	0.45	0.44	0.43	1

- Polyuria – 0.67
- Polydipsia – 0.65

Baseline Modeling

- Problem – classification
- Baseline model – Logistic Regression
- Null hypothesis – patient is NOT diabetic
- Goal – minimize Type II errors (False Negatives)
- Most important metric – Recall/Sensitivity (with respect to class '1')

Logistic Regression Performance

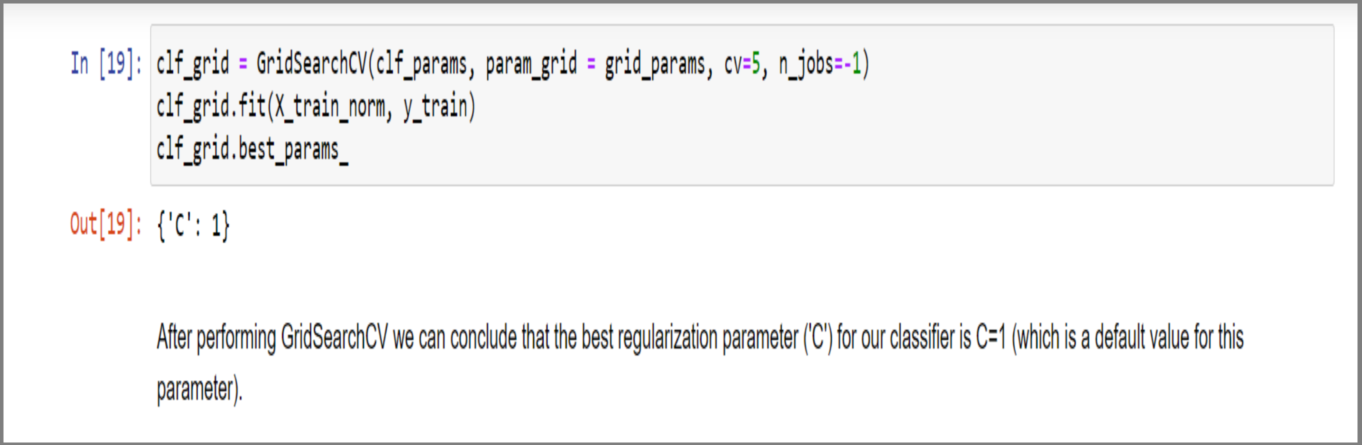


```
In [24]: print("=== Classification Report ===")
         print(classification_report(y_test, clfparams_ypred_test))
```

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

           0       0.89      0.97      0.93        40
           1       0.98      0.92      0.95        64

    accuracy                           0.94       104
   macro avg       0.93      0.95      0.94       104
weighted avg       0.95      0.94      0.94       104
```

- 92% of diabetic patients classified correctly
- False negatives – 8%

Logistic Regression Hyperparameters Tuning

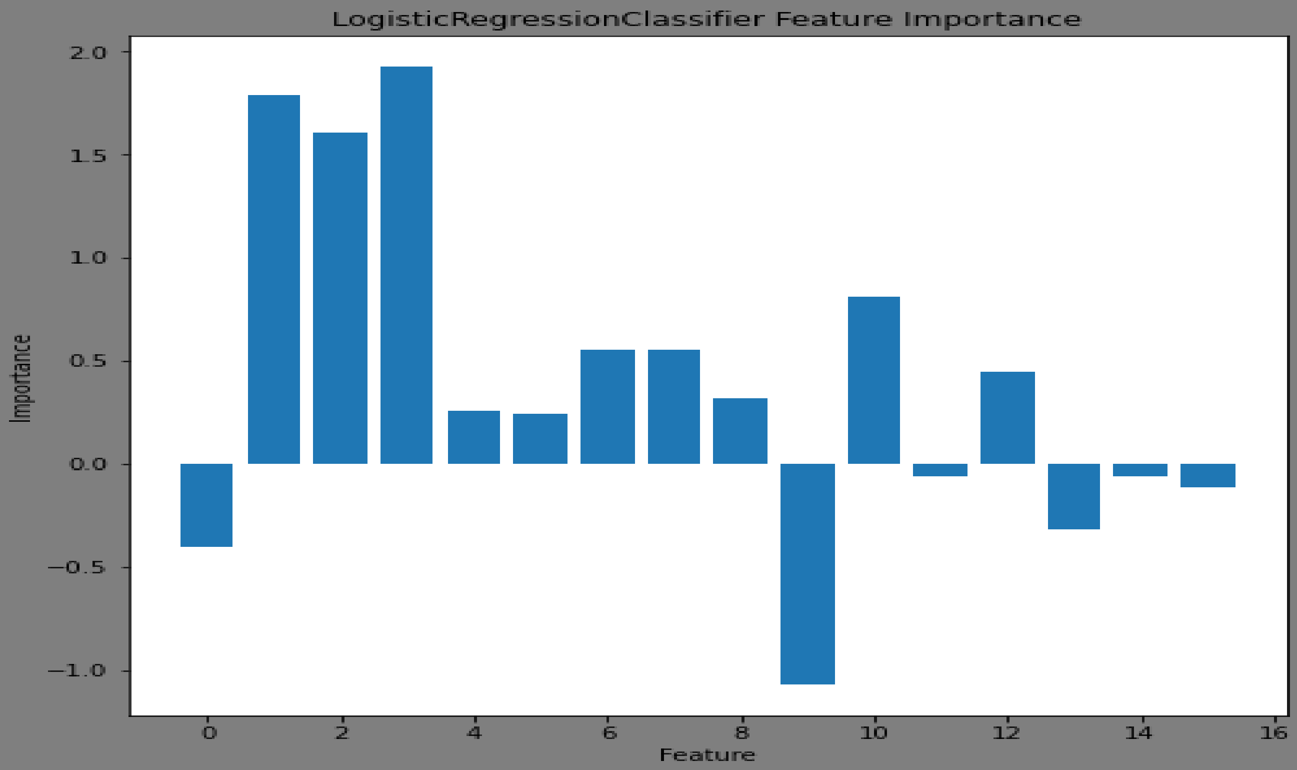


```
In [19]: clf_grid = GridSearchCV(clf_params, param_grid = grid_params, cv=5, n_jobs=-1)
         clf_grid.fit(X_train_norm, y_train)
         clf_grid.best_params_

Out[19]: {'C': 1}
```

After performing GridSearchCV we can conclude that the best regularization parameter ('C') for our classifier is C=1 (which is a default value for this parameter).

- Regularization parameter C = 1

Logistic Regression Feature Impact



- Positive – Polydipsia, Gender, Polyuria
- Negative – Itching, Age, Muscle Stiffness

Advanced Modeling

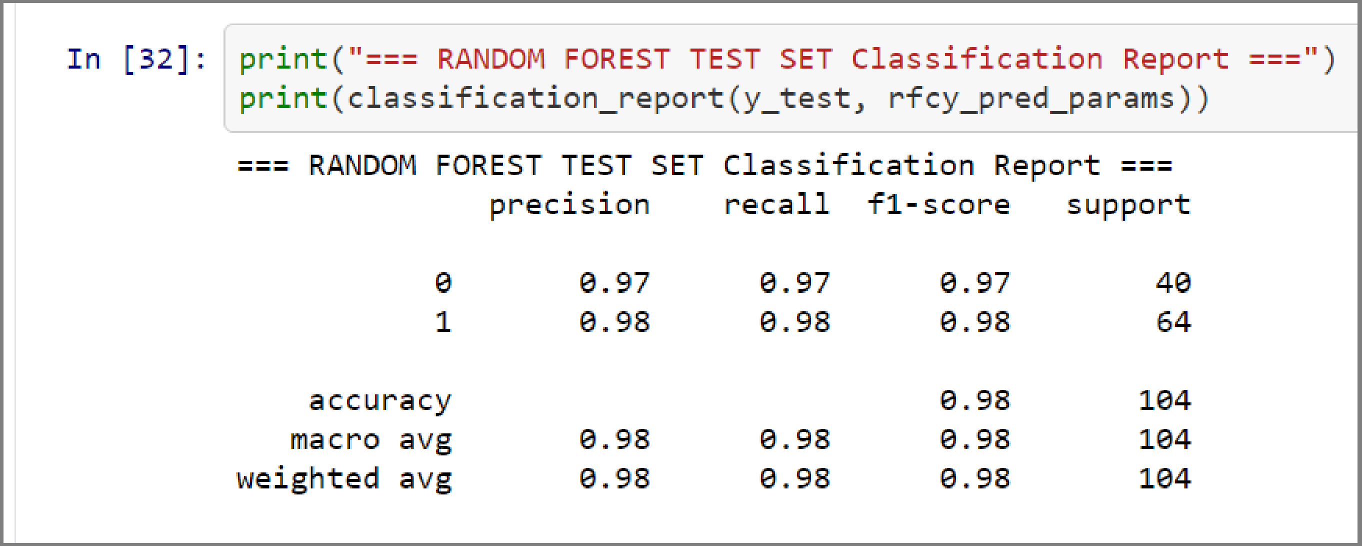
GOAL:

- Build additional models
- Compare their performance with baseline model

- MODELS:

- Random Forest
- Gradient Boosting
- Adaptive Boosting
- Support Vector Machine

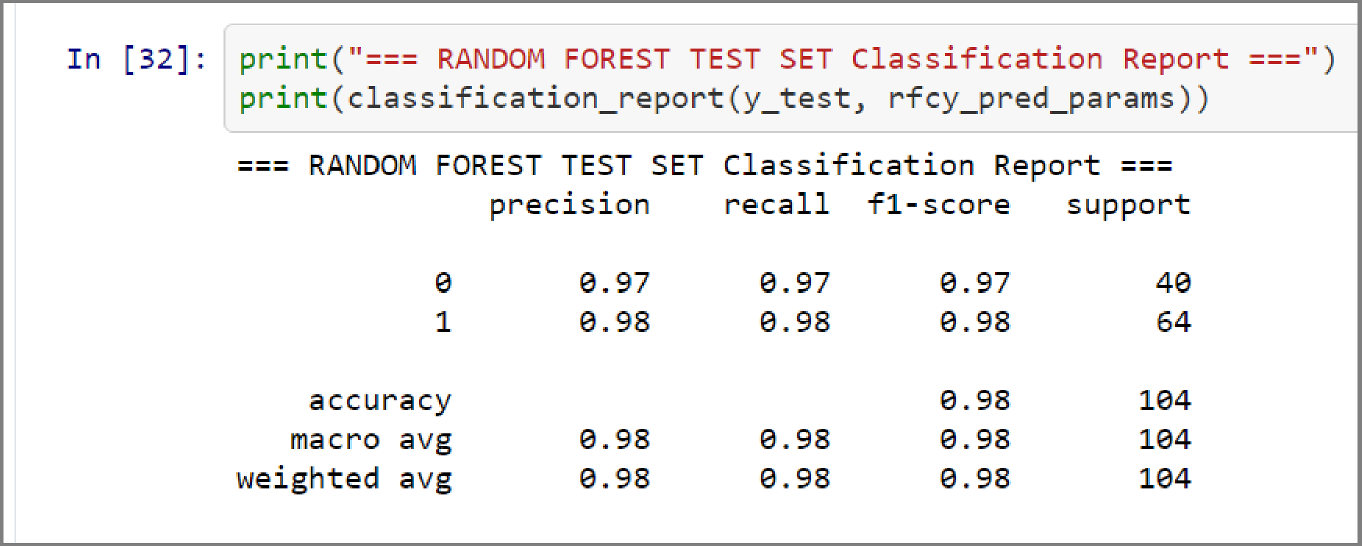
Random Forest Performance



```
In [32]: print("=== RANDOM FOREST TEST SET Classification Report ===")
         print(classification_report(y_test, rfcy_pred_params))

         === RANDOM FOREST TEST SET Classification Report ===
                       precision    recall  f1-score   support

                    0       0.97      0.97      0.97        40
                    1       0.98      0.98      0.98        64

             accuracy                           0.98       104
            macro avg       0.98      0.98      0.98       104
         weighted avg       0.98      0.98      0.98       104
```

- 98% of diabetic patients classified correctly
- False negatives – 2%

Gradient Boosting Performance

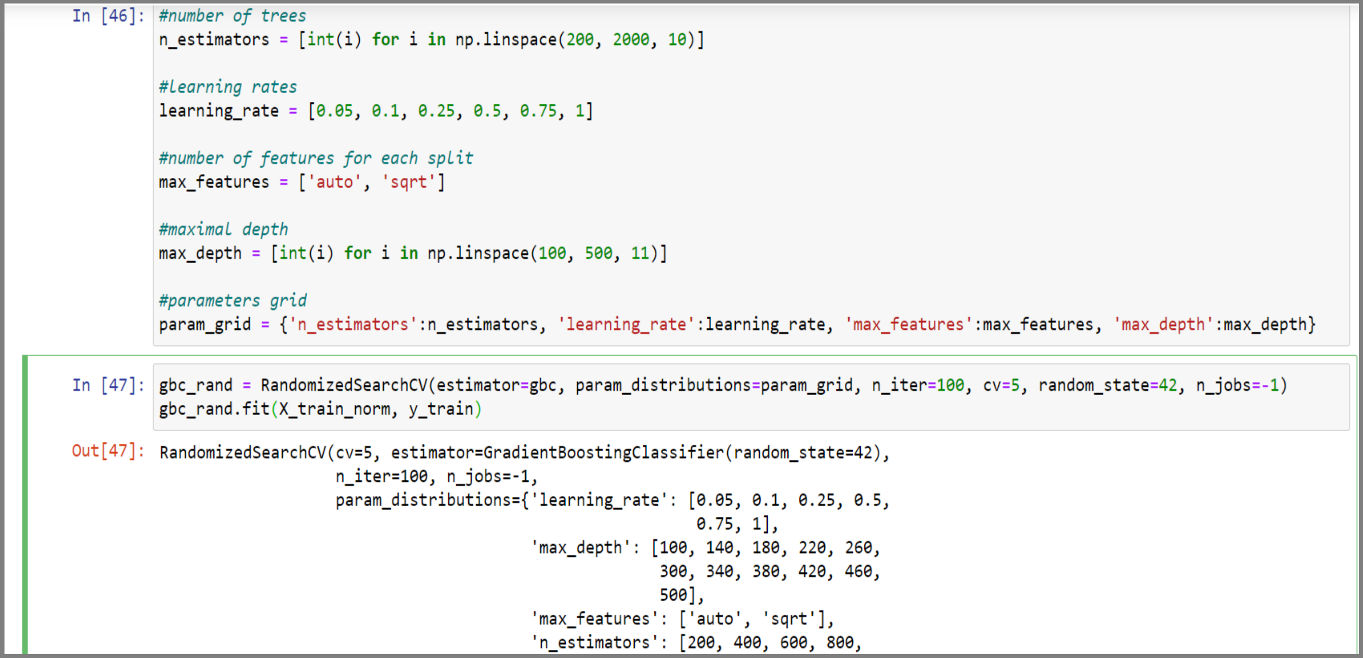


```
In [32]: print("=== RANDOM FOREST TEST SET Classification Report ===")
         print(classification_report(y_test, rfcy_pred_params))
```

```
=== RANDOM FOREST TEST SET Classification Report ===
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        40
           1       0.98      0.98      0.98        64

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104
```

- 98% of diabetic patients classified correctly
- False negatives – 2%

Gradient Boosting Hyperparameters Tuning

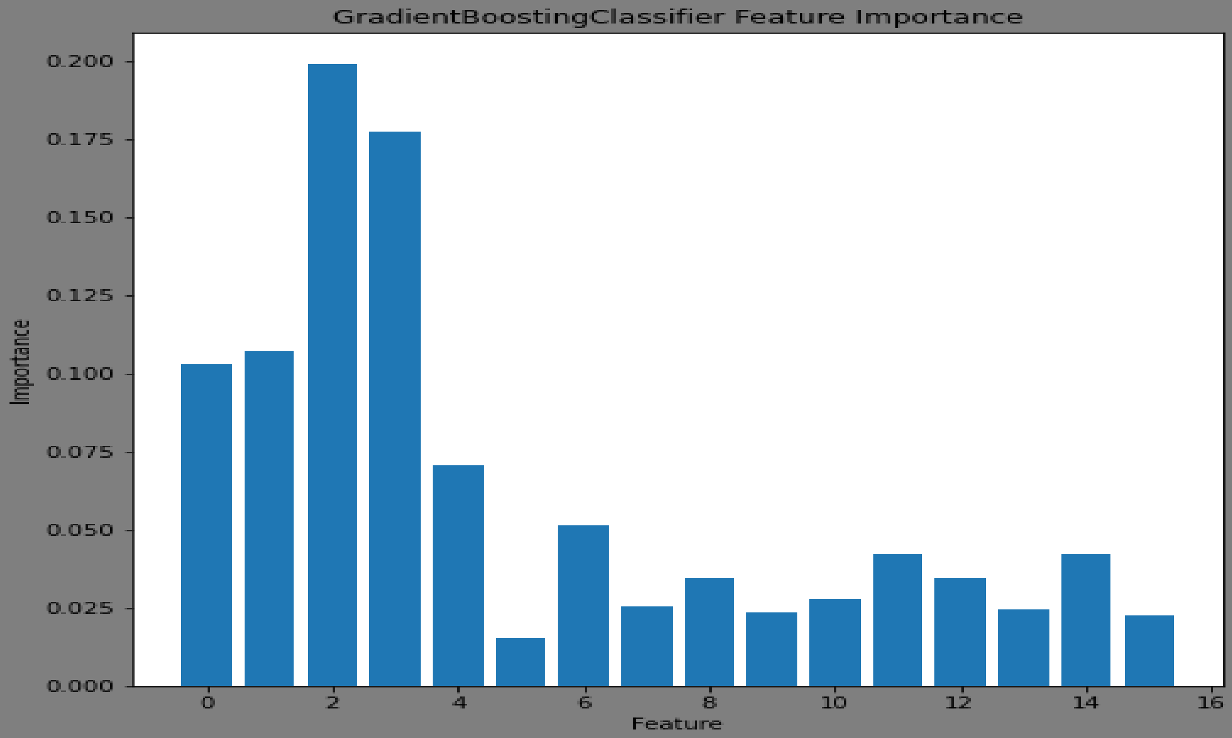


In [46]:
```
#number of trees
n_estimators = [int(i) for i in np.linspace(200, 2000, 10)]

#learning rates
learning_rate = [0.05, 0.1, 0.25, 0.5, 0.75, 1]

#number of features for each split
max_features = ['auto', 'sqrt']

#maximal depth
max_depth = [int(i) for i in np.linspace(100, 500, 11)]

#parameters grid
param_grid = {'n_estimators':n_estimators, 'learning_rate':learning_rate, 'max_features':max_features, 'max_depth':max_depth}
```

In [47]:
```
gbc_rand = RandomizedSearchCV(estimator=gbc, param_distributions=param_grid, n_iter=100, cv=5, random_state=42, n_jobs=-1)
gbc_rand.fit(X_train_norm, y_train)
```

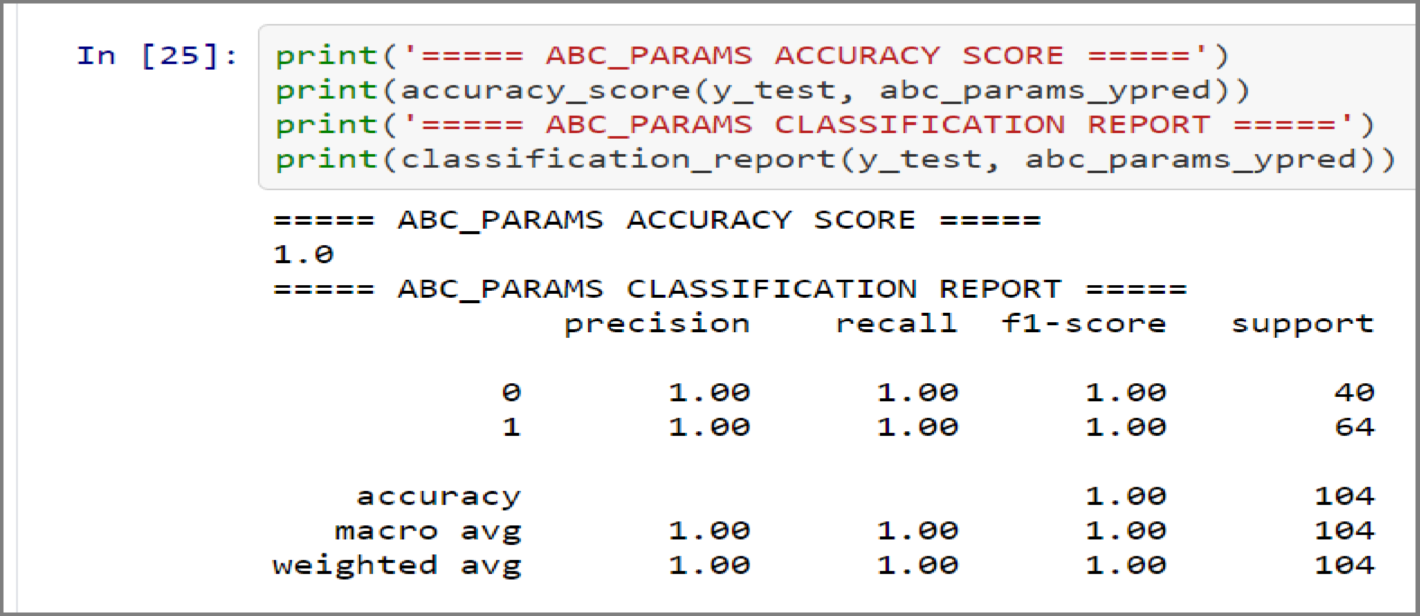
Out[47]:
```
RandomizedSearchCV(cv=5, estimator=GradientBoostingClassifier(random_state=42),
                   n_iter=100, n_jobs=-1,
                   param_distributions={'learning_rate': [0.05, 0.1, 0.25, 0.5,
                                                          0.75, 1],
                                        'max_depth': [100, 140, 180, 220, 260,
                                                      300, 340, 380, 420, 460,
                                                      500],
                                        'max_features': ['auto', 'sqrt'],
                                        'n_estimators': [200, 400, 600, 800,
```

Gradient Boosting Feature Importance



- Most important features:
  - Polyuria
  - Polydipsia
  - Gender

Adaptive Boosting Performance

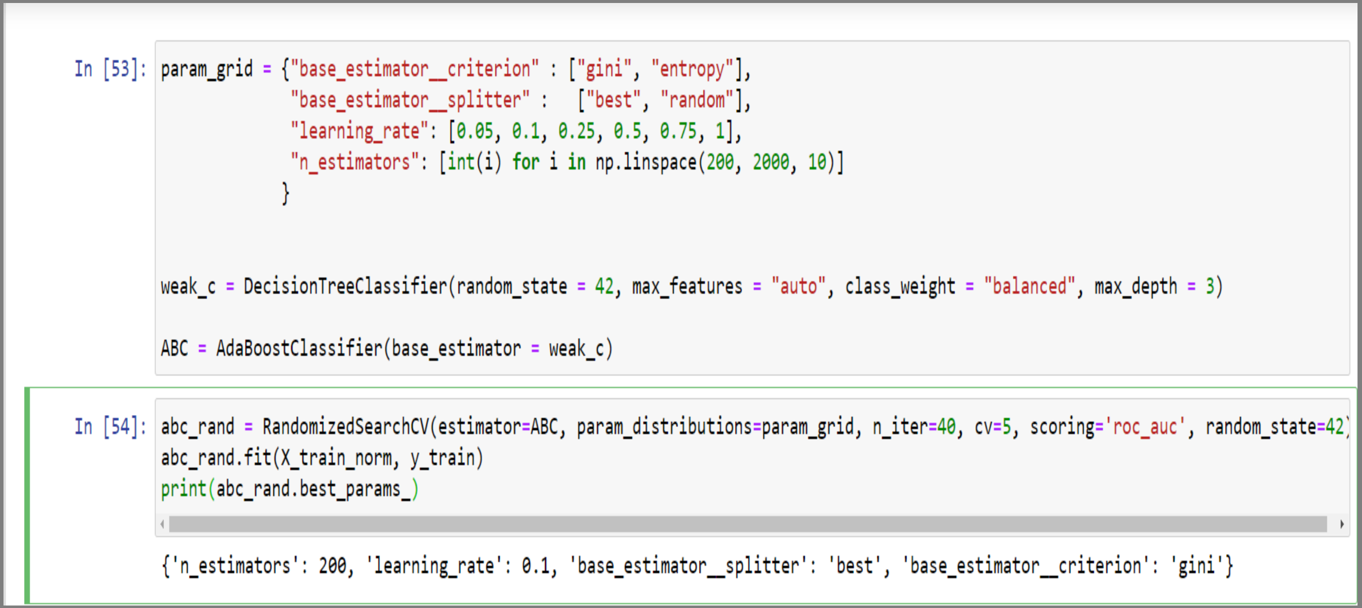


```
In [25]: print('===== ABC_PARAMS ACCURACY SCORE =====')
         print(accuracy_score(y_test, abc_params_ypred))
         print('===== ABC_PARAMS CLASSIFICATION REPORT =====')
         print(classification_report(y_test, abc_params_ypred))
```

```
===== ABC_PARAMS ACCURACY SCORE =====
1.0
===== ABC_PARAMS CLASSIFICATION REPORT =====
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        64

    accuracy                           1.00       104
   macro avg       1.00      1.00      1.00       104
weighted avg       1.00      1.00      1.00       104
```

- 100% of diabetic patients classified correctly
- False negatives – 0

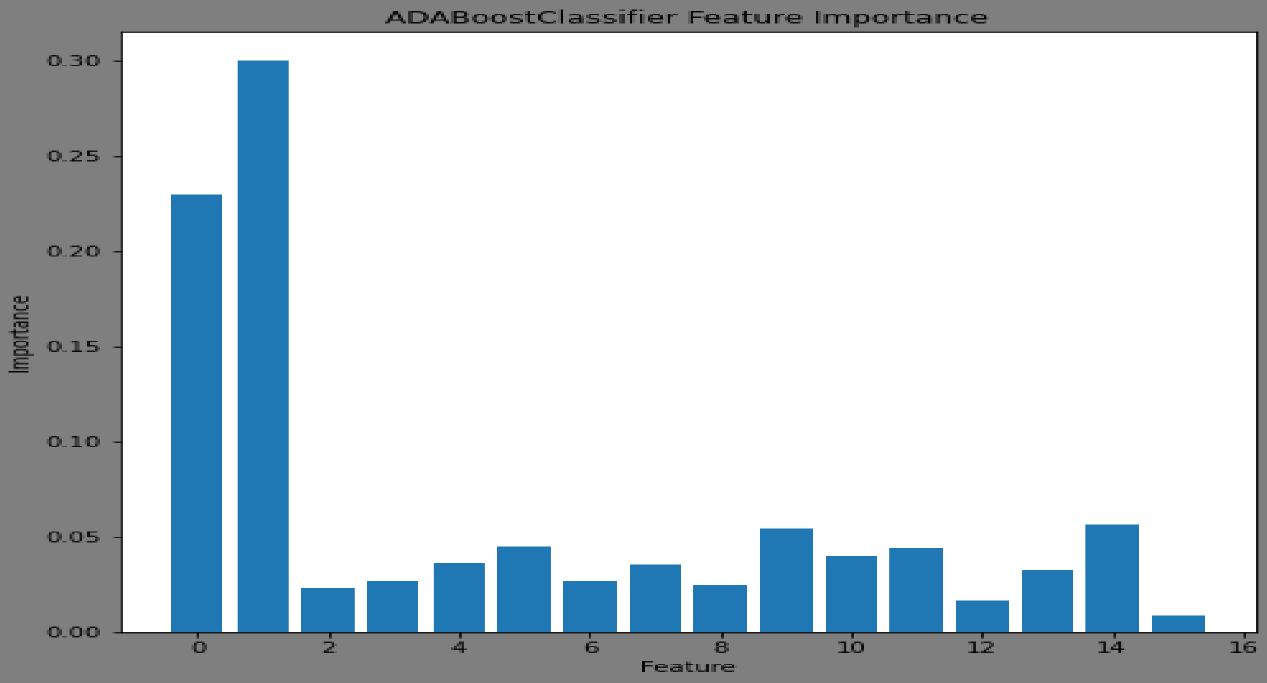
Adaptive Boosting Hyperparameter Tuning



```
In [53]: param_grid = {"base_estimator__criterion" : ["gini", "entropy"],
                       "base_estimator__splitter" :   ["best", "random"],
                       "learning_rate": [0.05, 0.1, 0.25, 0.5, 0.75, 1],
                       "n_estimators": [int(i) for i in np.linspace(200, 2000, 10)]
                      }


         weak_c = DecisionTreeClassifier(random_state = 42, max_features = "auto", class_weight = "balanced", max_depth = 3)

         ABC = AdaBoostClassifier(base_estimator = weak_c)
```

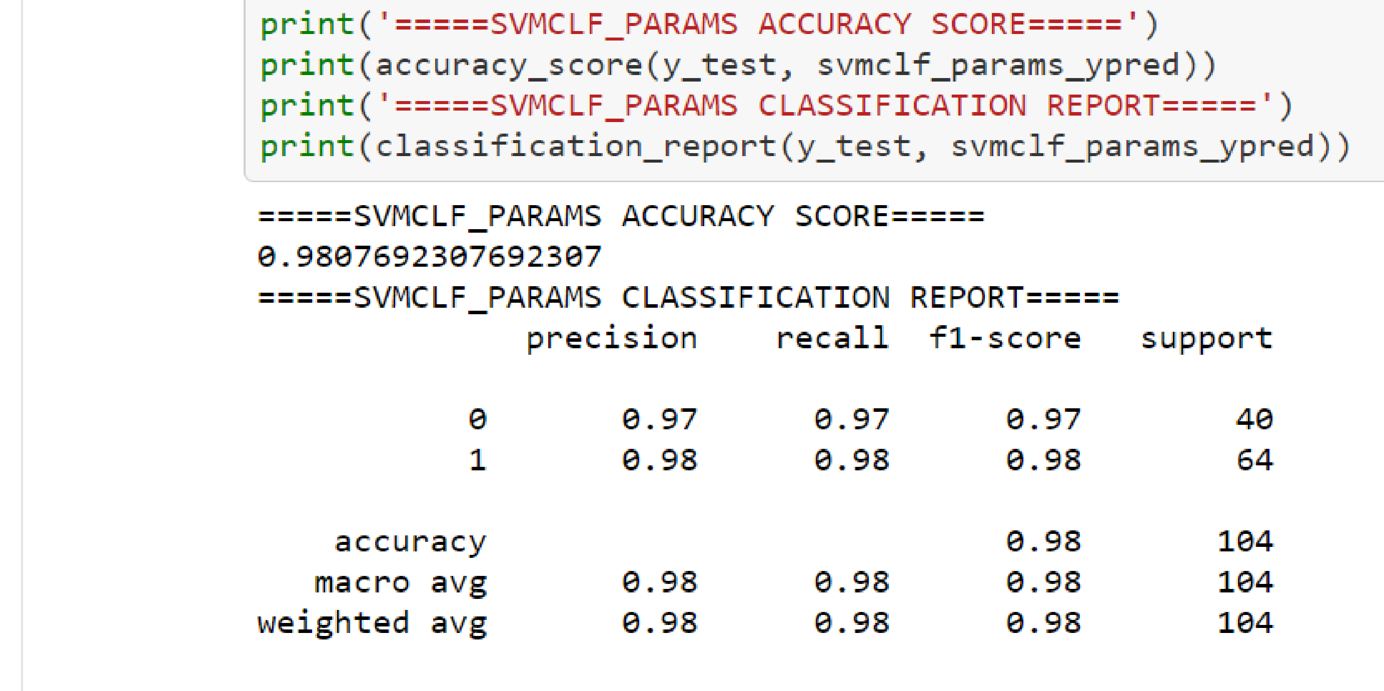
```
In [54]: abc_rand = RandomizedSearchCV(estimator=ABC, param_distributions=param_grid, n_iter=40, cv=5, scoring='roc_auc', random_state=42)
         abc_rand.fit(X_train_norm, y_train)
         print(abc_rand.best_params_)

         {'n_estimators': 200, 'learning_rate': 0.1, 'base_estimator__splitter': 'best', 'base_estimator__criterion': 'gini'}
```

Adaptive Boosting Feature Importance



- Most important features:
  - Age
  - Gender

Support Vector Machine Performance

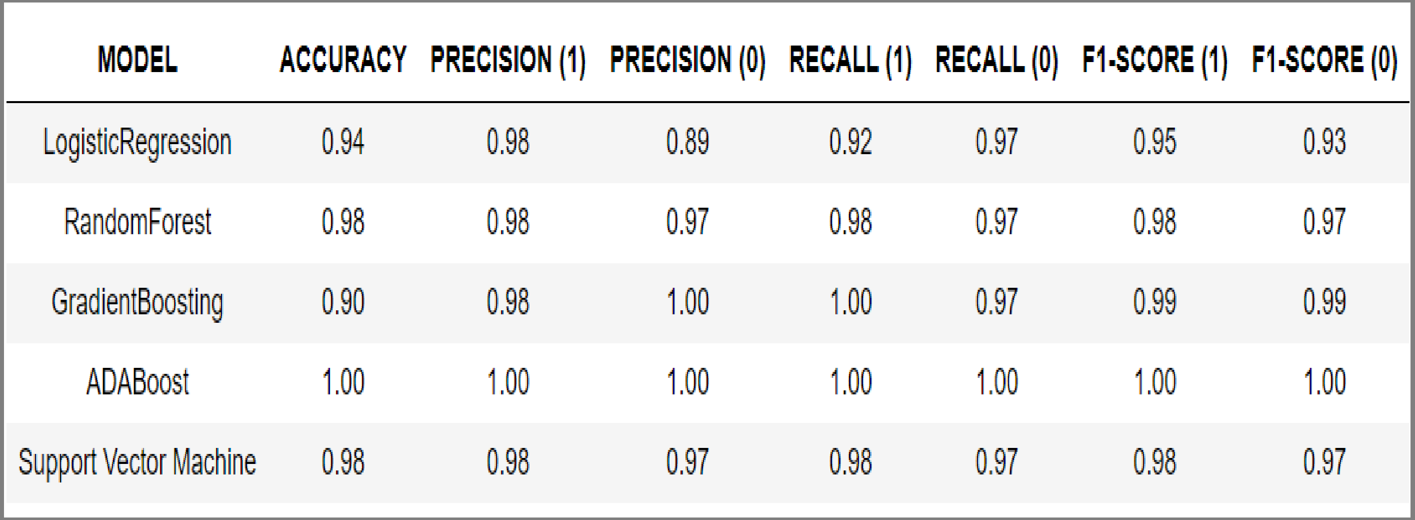


```
print('=====SVMCLF_PARAMS ACCURACY SCORE=====')
print(accuracy_score(y_test, svmclf_params_ypred))
print('=====SVMCLF_PARAMS CLASSIFICATION REPORT=====')
print(classification_report(y_test, svmclf_params_ypred))
```

```
=====SVMCLF_PARAMS ACCURACY SCORE=====
0.9807692307692307
=====SVMCLF_PARAMS CLASSIFICATION REPORT=====
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        40
           1       0.98      0.98      0.98        64

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104
```

- 98% of diabetic patients classified correctly
- False negatives – 2%

Modeling Results



MODEL	ACCURACY	PRECISION (1)	PRECISION (0)	RECALL (1)	RECALL (0)	F1-SCORE (1)	F1-SCORE (0)
LogisticRegression	0.94	0.98	0.89	0.92	0.97	0.95	0.93
RandomForest	0.98	0.98	0.97	0.98	0.97	0.98	0.97
GradientBoosting	0.90	0.98	1.00	1.00	0.97	0.99	0.99
ADABoost	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Support Vector Machine	0.98	0.98	0.97	0.98	0.97	0.98	0.97

- All models showed high scores
- Adaptive Boosting showed perfect scores (highly unusual)

Conclusions

- Tree-based models showed the best performance
- Adaptive boosting showed the best scores
- Adaptive boosting’s most important features are different from all other models
- Gradient Boosting showed second best score (98%)
- Gradient Boosting’s most important features are the same as all the other models’ (except ADABoost)

Future Work

- Train models with different train test split
- See if their performance changes significantly

Recommendations

- Implement Gradient Boosting as a production model
- Pay close attention to patients with Polyuria and Polydipsia
- Take into account that women seem to be more prone to diabetes than men