Diabetes Classification

IBM Machine Learning Professional Certificate
Supervised Machine Learning: Classification

Name – PAULAMI SANYAL

July 2023

OBJECTIVE

- The objective of this assignment is to predict the occurrence of Diabetes, based on diagnostic measurements of many previous cases.
- Our model will try to learn from the available records of the previously diagnosed patients and classify new unknown patients with similar records, whether they have diabetes or not, based on those features.

DATA SUMMARY

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. We have 768 datapoints and 8 features and one target variable 'Outcome'.

- Pregnancies: Number of times the women have been pregnant before
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test (normal level - 100 mg/dL during fasting and less than 140 mg/dL 2-hours postprandial)
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm). If you've had diabetes for a long time without good control of your blood sugar, you could develop this condition. Poor blood supply to the skin can cause changes in the collagen and fat underneath. The overlaying skin becomes thin and red.
- Insulin: 2-Hour serum insulin (mu U/ml). Used to assess how an individual processes glucose and how the insulin in the body responds to those glucose levels.
- BMI: Body mass index of the person(weight in kg/(height in m)^2). Any increase in BMI above normal weight levels is associated with an increased risk of being diagnosed as having complications of diabetes mellitus.
- DiabetesPedigreeFunction: Indicates the function which scores likelihood of diabetes based on family history.
- Age: Age (in years)
- Outcome: Class variable (0 for patients without diabetes or 1 for people with diabetes)

DATA EXPLORATION

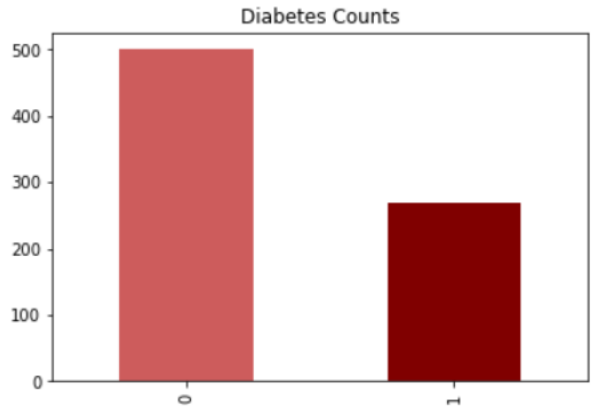
- In our dataset, we have 8 features and one target variable ‘Outcome’. The target variable ‘Outcome’ is classified into 2 categories, 1 for those patients who has diabetes and 0 for those patients who do not have diabetes.

- Count-wise, 500 patients do not have diabetes and 268 patients do have diabetes. Percent-wise, about 65% do not have diabetes, and about 35% have diabetes.

```
print(df['Outcome'].value_counts())
print(df['Outcome'].value_counts(normalize=True))

df['Outcome'].value_counts().plot(kind='bar', title='Diabetes Counts', color=['indianred', 'maroon'])
plt.show()
```

```
0    500
1    268
Name: Outcome, dtype: int64
0    0.651042
1    0.348958
Name: Outcome, dtype: float64
```

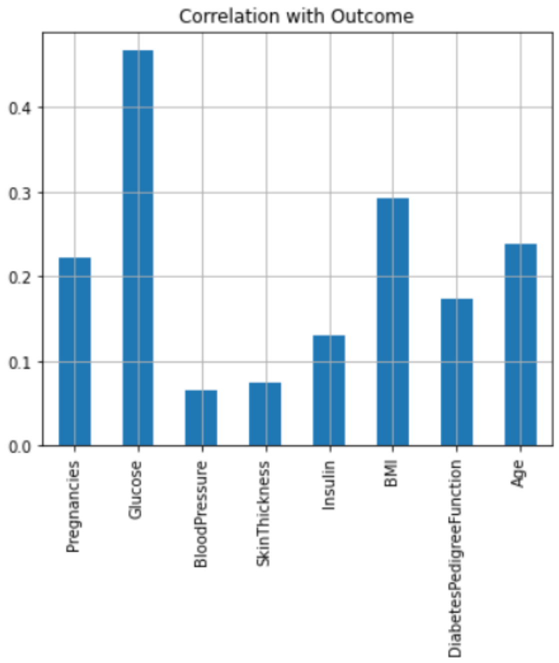


DATA EXPLORATION

As we can see from the bar plot of the correlations of the features with the ‘Outcome’, the features that have highest correlation with the target are ‘Glucose’, BMI’, ‘Age’, and ‘Pregnancies’.

```
df.drop('Outcome', axis=1).corrwith(df.Outcome).plot(kind='bar', grid=True, figsize=(6, 5), title="Correlation with Outcome")
```

```
<AxesSubplot:title={'center':'Correlation with Outcome'}>
```

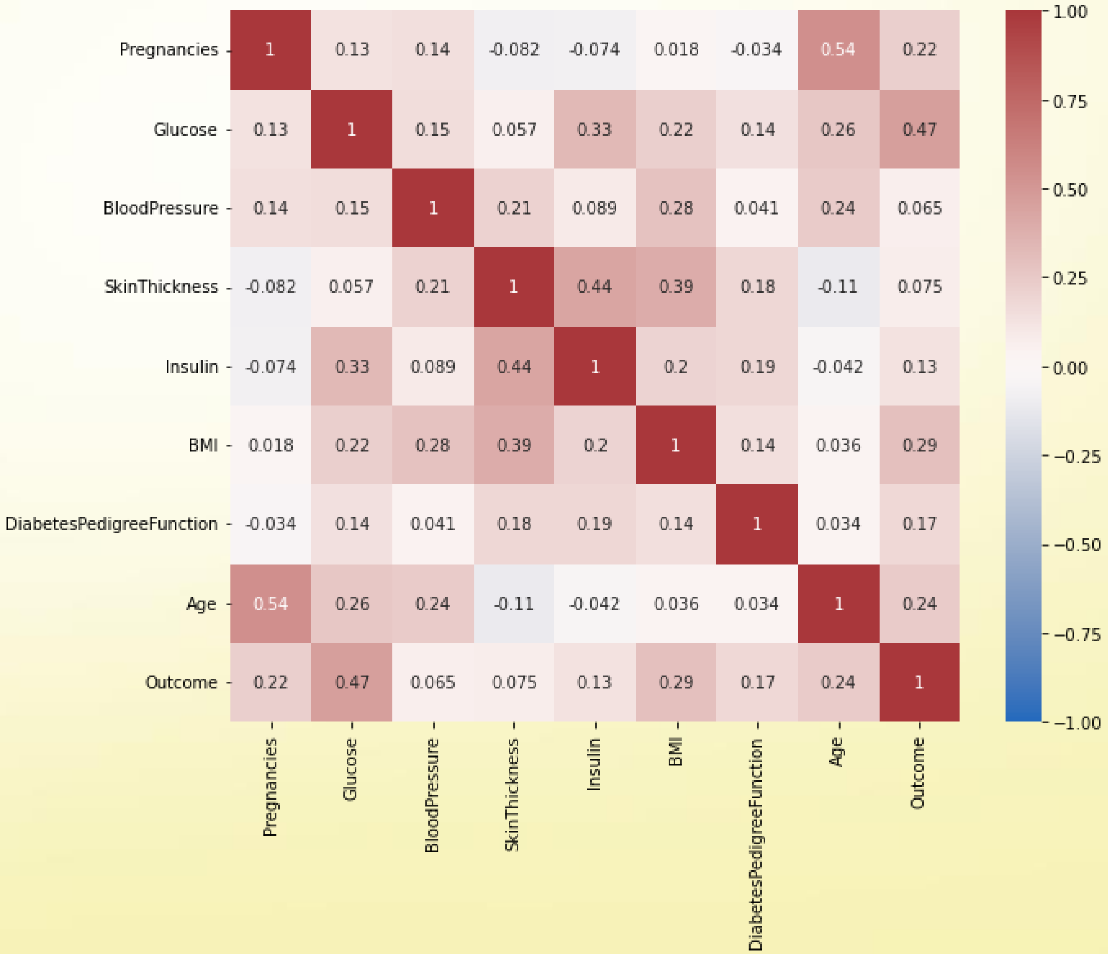


DATA EXPLORATION

- Heatmaps are used to find correlations between all the feature variables in the dataset.

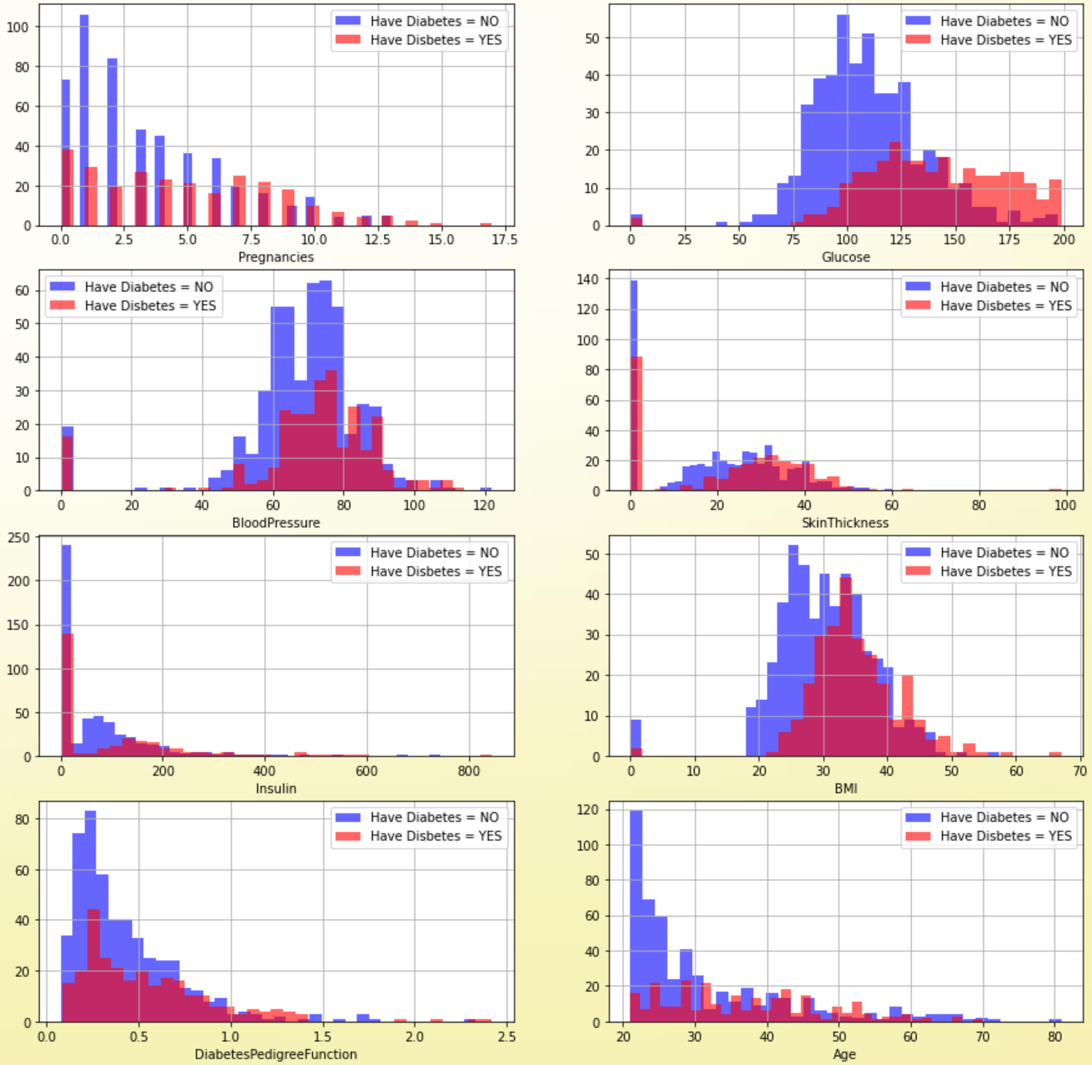
- As we can see, ‘Glucose’ and ‘BMI’ both have higher correlations with our target variable ‘Outcome’.

- ‘SkinThickness’ and ‘Age’ have negative correlation. Meaning, as there is an increase in age, there’s a decrease in ‘SkinThickness’.



DATA EXPLORATION

Here we see the distributions of occurrence of diabetes amongst various given features.



DATA CLEANING & FEATURE ENGINEERING

- The dataset came with 8 features. And all of them were continuous in nature.
- However, some of the features had much higher values than the others. Hence, we had to normalize the data.
- We normalized the data after splitting it, and performed **fit_transform()** function only on the **X_train**, and then performed only **transform()** function on **X_test** to avoid overfitting.

CLASSIFIER MODELS

LOGISTIC REGRESSION - We built 3 models with Logistic Regression – a standard one without regularization, one with L1 regularization and one with L2 regularization. All the models gave us similar scores. So, it's safe to say here regularization isn't really required.

```python
# Standard logistic regression
lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
y_pred_0 = lr.predict(X_test)
clf_report = pd.DataFrame(classification_report(y_test, y_pred_0, output_dict=True))
clf_report
```

	0	1	accuracy	macro avg	weighted avg
precision	0.769231	0.677419	0.744589	0.723325	0.737037
recall	0.866667	0.518519	0.744589	0.692593	0.744589
f1-score	0.815047	0.587413	0.744589	0.701230	0.735227
support	150.000000	81.000000	0.744589	231.000000	231.000000

```python
# L1 regularized logistic regression
lr_l1 = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear').fit(X_train, y_train)
y_pred_1 = lr_l1.predict(X_test)
clf_report = pd.DataFrame(classification_report(y_test, y_pred_1, output_dict=True))
clf_report
```

	0	1	accuracy	macro avg	weighted avg
precision	0.769231	0.677419	0.744589	0.723325	0.737037
recall	0.866667	0.518519	0.744589	0.692593	0.744589
f1-score	0.815047	0.587413	0.744589	0.701230	0.735227
support	150.000000	81.000000	0.744589	231.000000	231.000000

```python
# L2 regularized logistic regression
lr_l2 = LogisticRegressionCV(Cs=10, cv=4, penalty='l2', solver='liblinear').fit(X_train, y_train)
y_pred_2 = lr_l2.predict(X_test)
clf_report = pd.DataFrame(classification_report(y_test, y_pred_2, output_dict=True))
clf_report
```

	0	1	accuracy	macro avg	weighted avg
precision	0.769231	0.677419	0.744589	0.723325	0.737037
recall	0.866667	0.518519	0.744589	0.692593	0.744589
f1-score	0.815047	0.587413	0.744589	0.701230	0.735227
support	150.000000	81.000000	0.744589	231.000000	231.000000

CLASSIFIER MODELS

KNN Algorithm – We also built a KNN model on the data.

```
knn = KNeighborsClassifier(n_neighbors=25, weights='distance')
knn = knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

KNN_report = pd.DataFrame(classification_report(y_test, y_pred, output_dict=True))
KNN_report
```

	0	1	accuracy	macro avg	weighted avg
precision	0.751412	0.685185	0.735931	0.718299	0.728190
recall	0.886667	0.456790	0.735931	0.671728	0.735931
f1-score	0.813456	0.548148	0.735931	0.680802	0.720426
support	150.000000	81.000000	0.735931	231.000000	231.000000

CLASSIFIER MODELS

XGBoost Classifier Algorithm – We also used XGBoost as one of the models for the dataset.

```
final_xgb_cl = xgb.XGBClassifier(
    **grid_cv.best_params_,
    objective="binary:logistic",
)

_ = final_xgb_cl.fit(X_train, y_train)

y_pred = final_xgb_cl.predict(X_test)

xgb_report = pd.DataFrame(classification_report(y_test, y_pred, output_dict=True))
xgb_report
```

	0	1	accuracy	macro avg	weighted avg
precision	0.767956	0.780000	0.770563	0.773978	0.772179
recall	0.926667	0.481481	0.770563	0.704074	0.770563
f1-score	0.839879	0.595420	0.770563	0.717650	0.754160
support	150.000000	81.000000	0.770563	231.000000	231.000000

CLASSIFIER MODELS

SVC algorithm – We built a support vector classifier model as well for the dataset.

```
from sklearn.svm import SVC

kwargs = {'kernel': 'rbf'}
svc = SVC(**kwargs)

SVC_cl = svc.fit(X_train, y_train)
y_pred = SVC_cl.predict(X_test)
SVC_cl_report = pd.DataFrame(classification_report(y_test, y_pred, output_dict=True))
SVC_cl_report
```

	0	1	accuracy	macro avg	weighted avg
precision	0.778443	0.687500	0.753247	0.732972	0.746554
recall	0.866667	0.543210	0.753247	0.704938	0.753247
f1-score	0.820189	0.606897	0.753247	0.713543	0.745398
support	150.000000	81.000000	0.753247	231.000000	231.000000

RECOMMENDATION

- If we are making classification of diseases, Recall (or Sensitivity) would be the ideal score to use as a metric for decision-making.
- In our models, we find that Support Vector Classifier gave us the highest Recall value of about 54%.
- The close second would be Logistic Regression, with or without any regularization.
- My recommendation would be to use Support Vector Classifier in this case.

SUGGESTIONS/NEXT STEPS

- From the metrics obtained from the model classification report, our Recall scores were quite low. A recall value of less than 0.5 could be due to imbalanced class or untuned hyperparameters. We can further inspect the features for class imbalance and perform resampling, oversampling or undersampling or simply using class_weight as a parameter in the models, wherever applicable.
- Various other ensemble methods can also be used find the best model fit for this dataset.

THANK YOU!

Source: kaggle.com