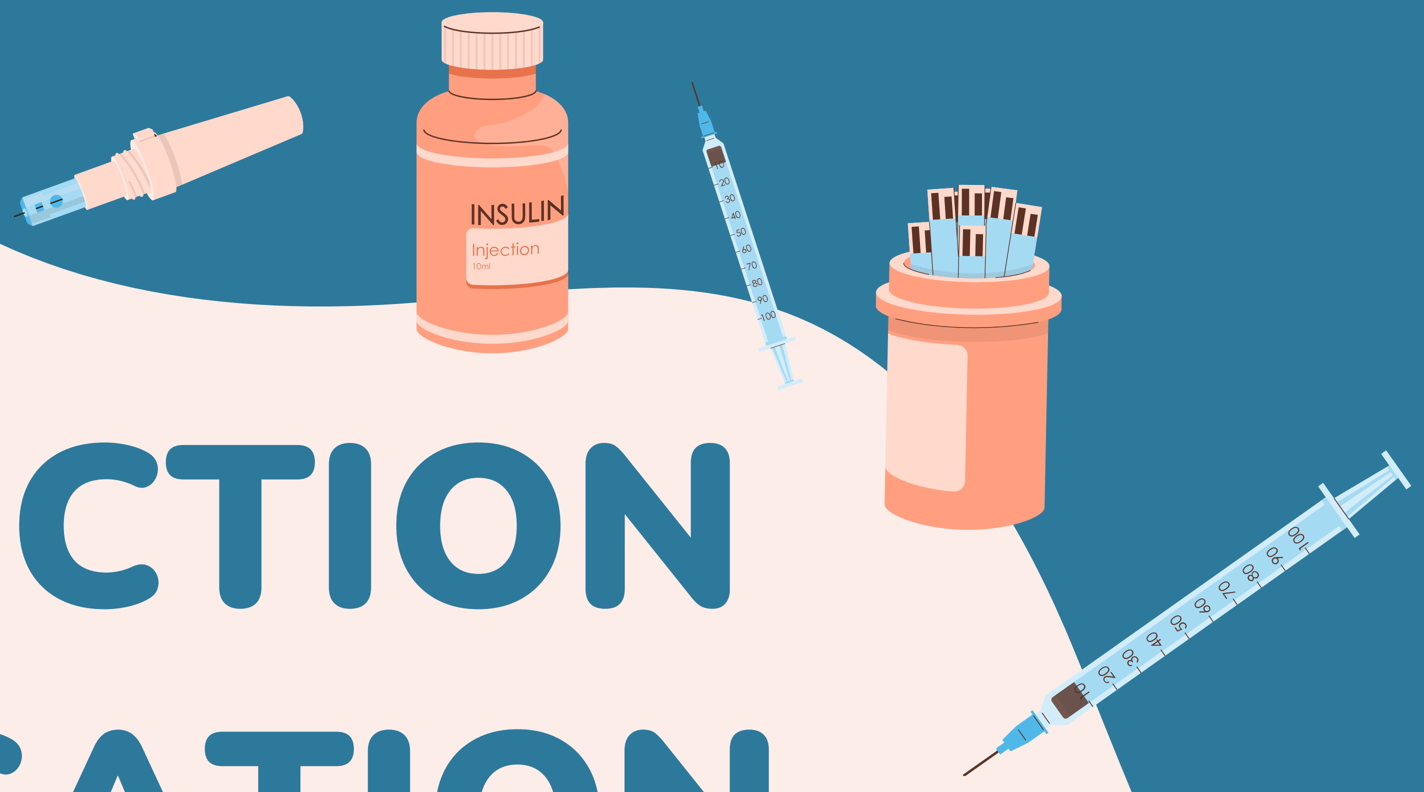


DIABETES PREDICTION USING CLASSIFICATION

Created by:
Lin Xinyang
Shivam Varshney (456533)

OVERVIEW

1. Project description
2. Data preprocessing
3. Data visualization
4. Building models
5. performance comparison
6. further iprovement.

1.Project Description

- **Objective:** To leverage Classification Algorithm to predict Diabetes in female patients.
- **Datasource:** Prima Indians Database by National Institute of Diabetes and Digestive and Kidney Diseases
- **Tools:** Five classification algorithms including Random ForestClassifier, AdaBoostClassifier, XGBClassifier, SVM, LightBGM; one regression algorithm logistic regression.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	[illegible]
5	5	116	74	0	0	25.6	0.201	30	[illegible]
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

- **Source of the data:** The National Institute of Diabetes and Digestive and Kidney Diseases

2.Data Preprocessing

Data cleaning

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
```

```
Data columns (total 9 columns):
 #   Column                    Non-Null Count
---  ------                    -------------
 0   Pregnancies               768 non-null
 1   Glucose                   768 non-null
 2   BloodPressure             768 non-null
 3   SkinThickness             768 non-null
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

```python
# Replace missing values with column medians
diabetes_df_copy = diabetes_df.copy(deep = True)
columns_median=np.array([np.median(diabetes_df_copy["Glucose"]),
                         np.median(diabetes_df_copy["BloodPressure"]),
                         np.median(diabetes_df_copy["SkinThickness"]),
                        np.median(diabetes_df_copy["Insulin"]),
                         np.median(diabetes_df_copy["BMI"])])

diabetes_df_copy.loc[diabetes_df_copy["Glucose"]==0,"Glucose"]= columns_median[0]
diabetes_df_copy.loc[diabetes_df_copy["BloodPressure"]==0,"BloodPressure"]= columns_median[1]
diabetes_df_copy.loc[diabetes_df_copy["SkinThickness"]==0,"SkinThickness"]= columns_median[2]
diabetes_df_copy.loc[diabetes_df_copy["Insulin"]==0,"Insulin"]= columns_median[3]
```

Data Preprocessing

```
# 1. BMI as a Range
diabetes_df_copy['BMI_Category'] = pd.cut(diabetes_df_copy['BMI'], bins=[0, 18.5, 24.9, 29.9, 34.9, 39.9,
                                    labels=['Underweight', 'Normal Weight', 'Overweight', 'Obese I', 'Ob

# 2. Interaction Term: Glucose * Insulin
diabetes_df_copy['Glucose_Insulin_Interact'] = diabetes_df_copy['Glucose'] * diabetes_df_copy['Insulin']

# 3. Age Groups
diabetes_df_copy['Age_Group'] = pd.cut(diabetes_df_copy['Age'], bins=[20, 30, 40, 50, 60, 70, np.inf],
                                labels=['20-30', '30-40', '40-50', '50-60', '60-70', '70+'])
diabetes_df_copy
```

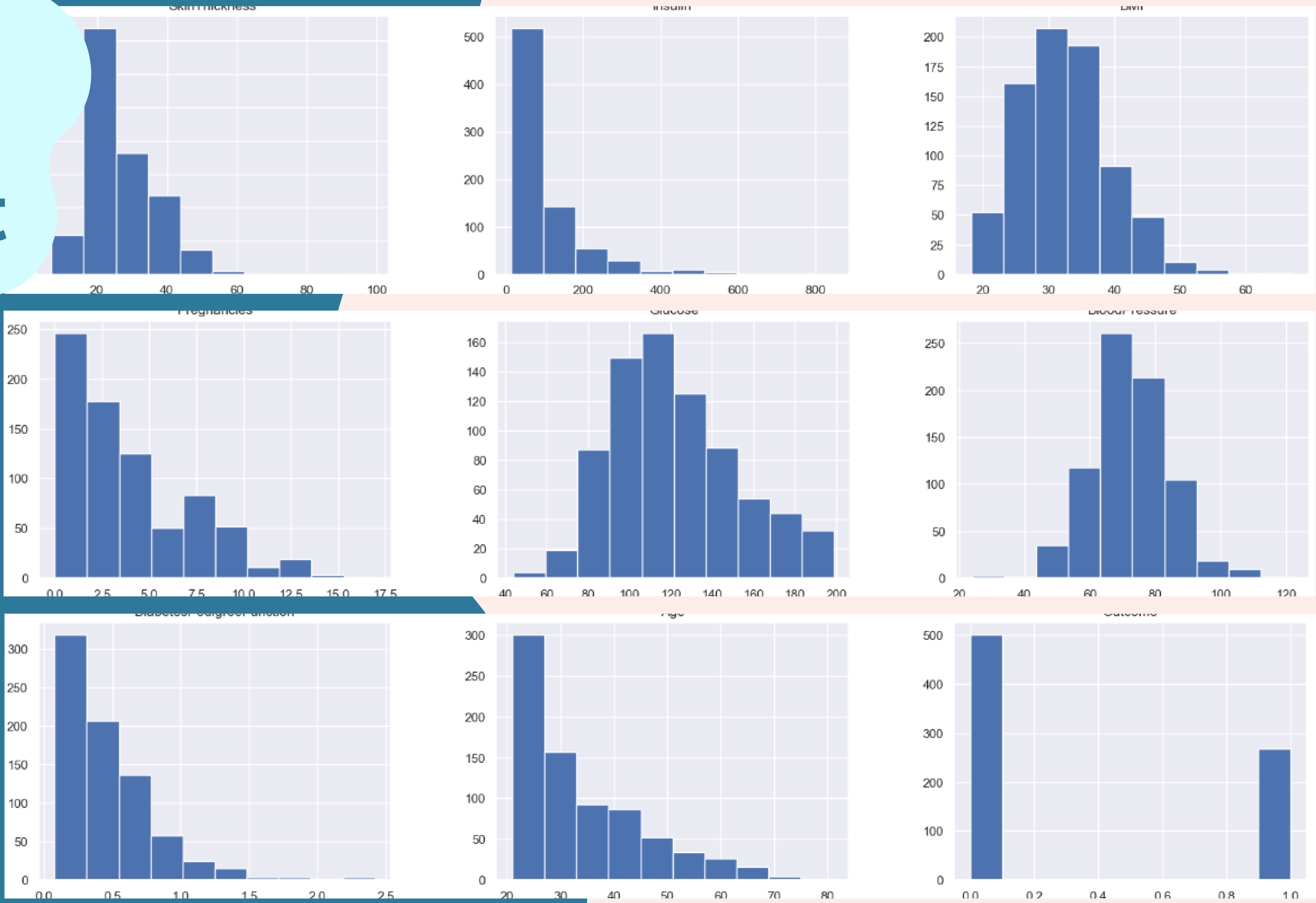
Newly created features:

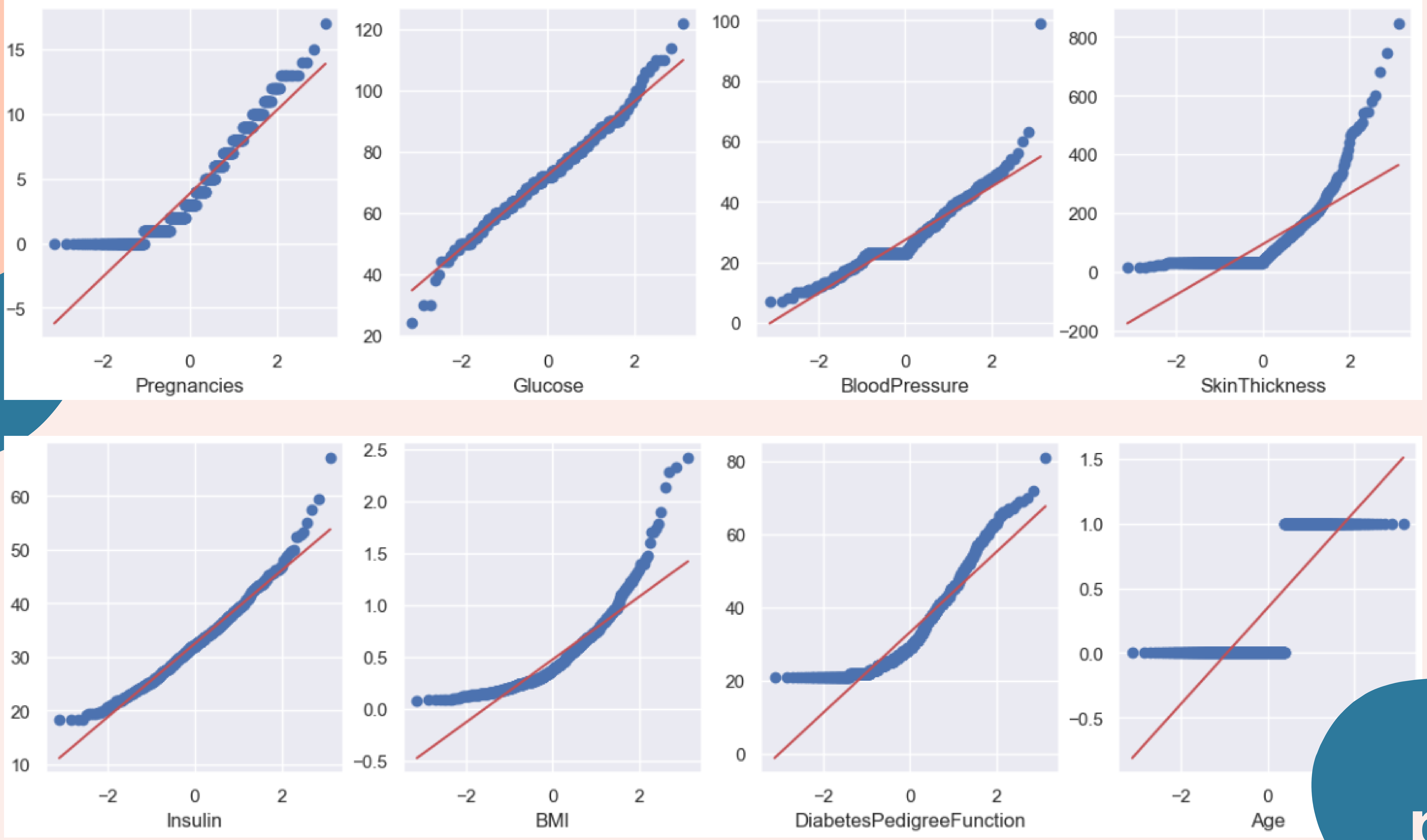
- **BMI_Category** (categorical)
- **Age_Group** (categorical)
- **Glucose_Insulin_Interact** (numerical)

Data engineering

3.Data Visualization

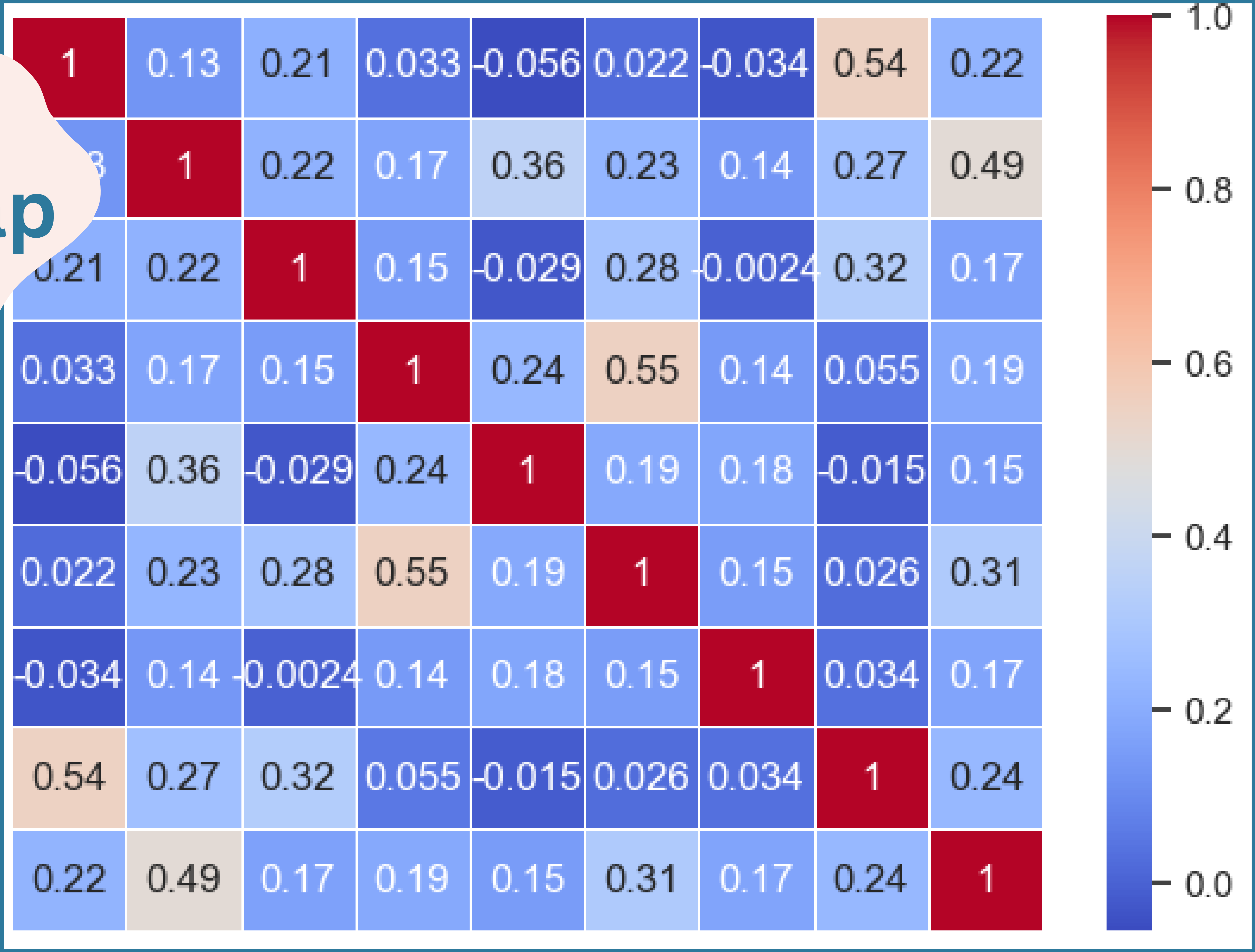
Bar chart



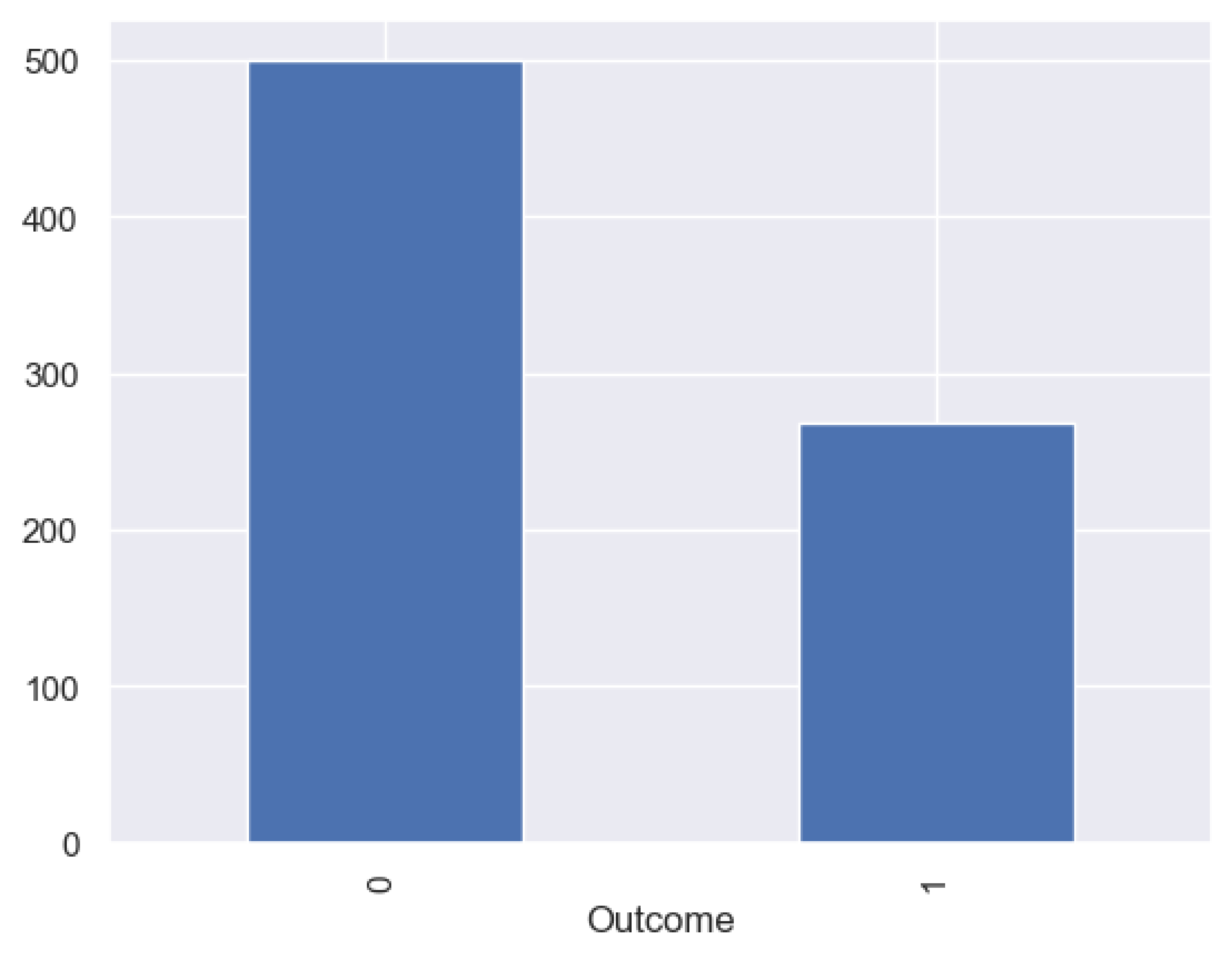

Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age

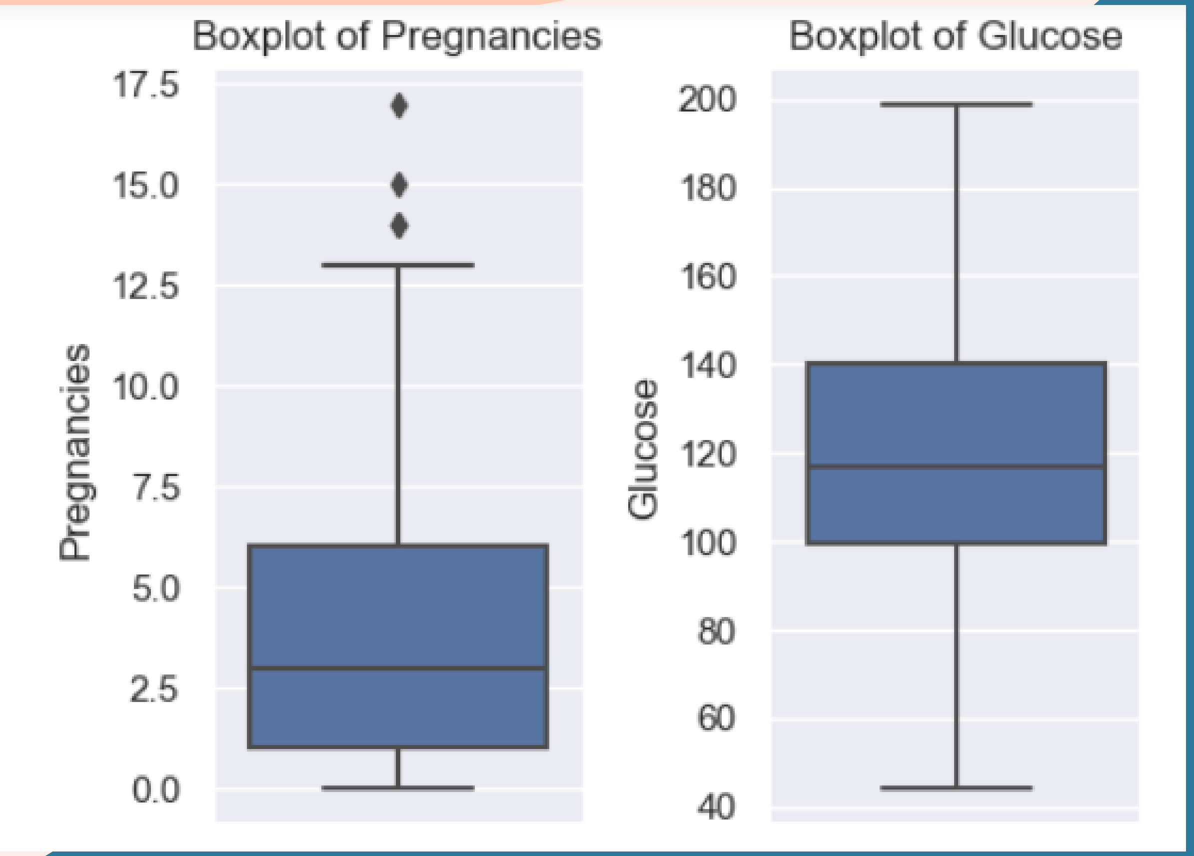
QQ plots

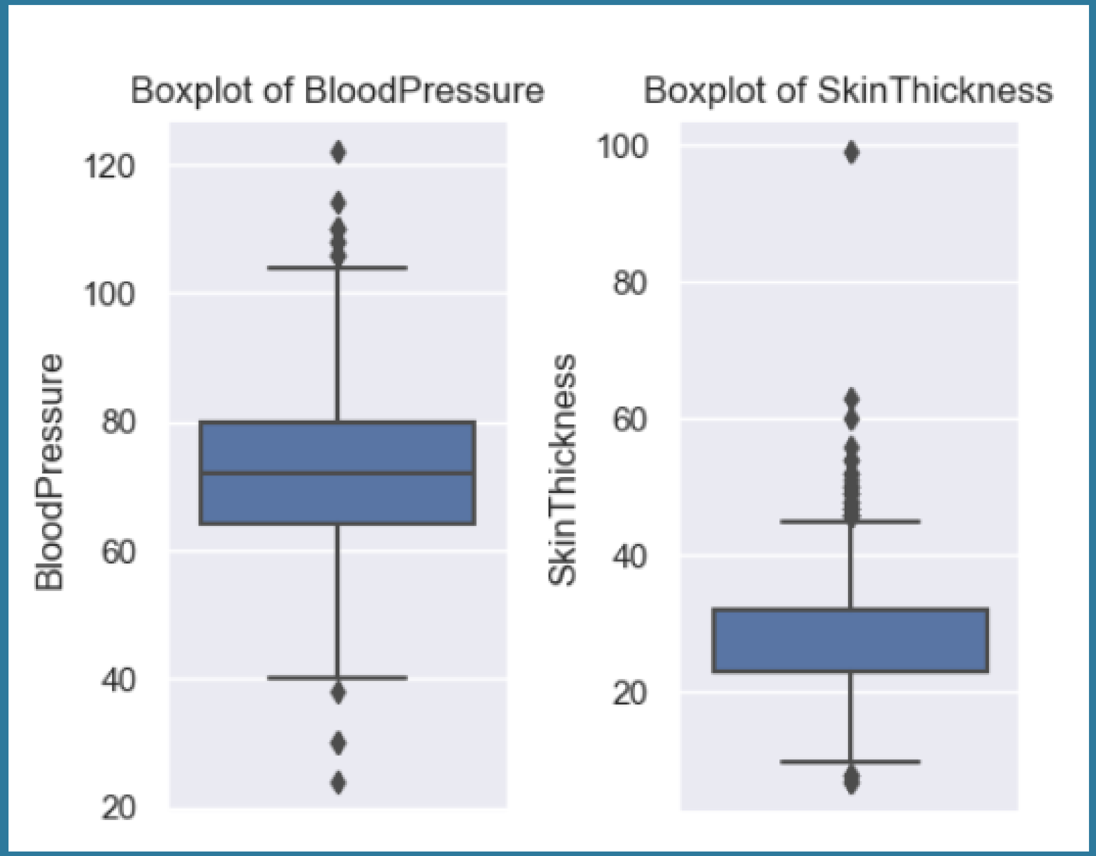
Heatmap

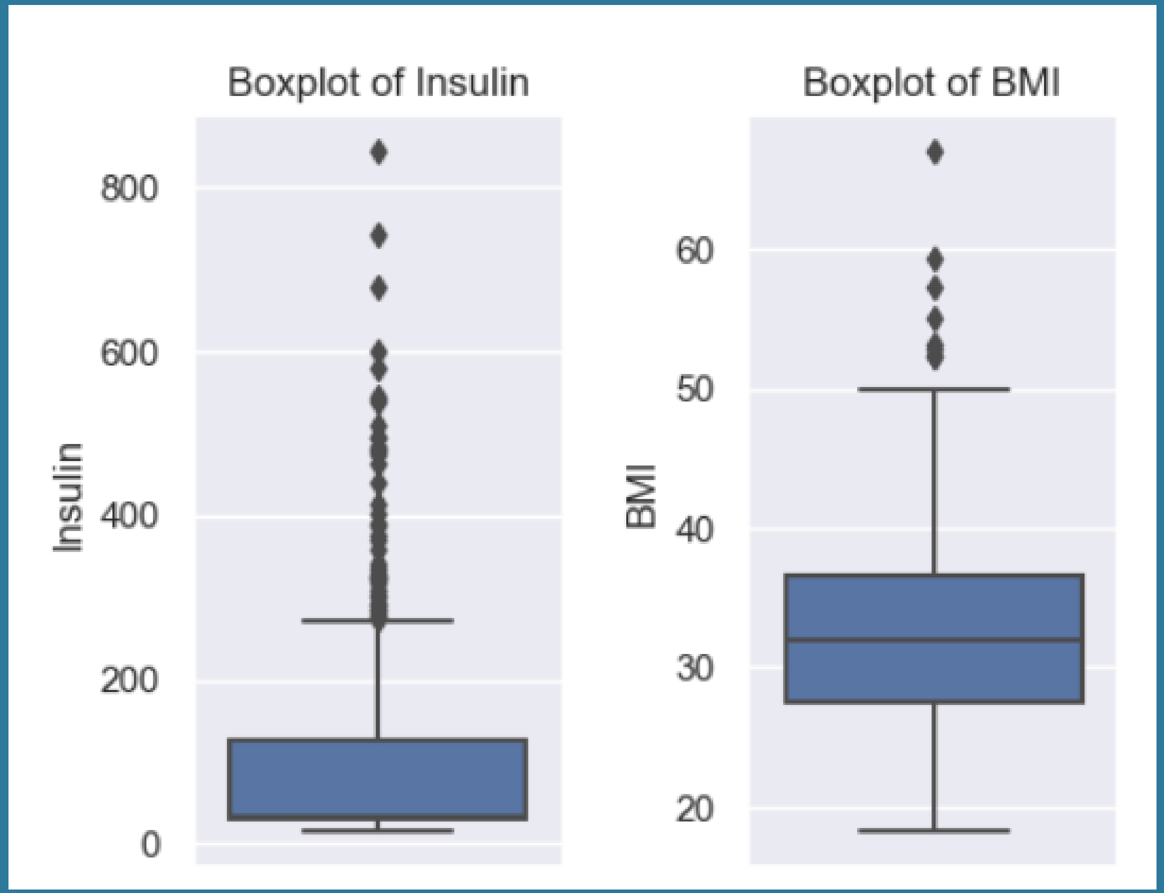

1 0.13 0.21 0.033 -0.056 0.022 -0.034 0.54 0.22
1 0.22 0.17 0.36 0.23 0.14 0.27 0.49
0.21 0.22 1 0.15 -0.029 0.28 -0.0024 0.32 0.17
0.033 0.17 0.15 1 0.24 0.55 0.14 0.055 0.19
-0.056 0.36 -0.029 0.24 1 0.19 0.18 -0.015 0.15
0.022 0.23 0.28 0.55 0.19 1 0.15 0.026 0.31
-0.034 0.14 -0.0024 0.14 0.18 0.15 1 0.034 0.17
0.54 0.27 0.32 0.055 -0.015 0.026 0.034 1 0.24
0.22 0.49 0.17 0.19 0.15 0.31 0.17 0.24 1
1.0
0.8
0.6
0.4
0.2
0.0

Checking the Balance of our data

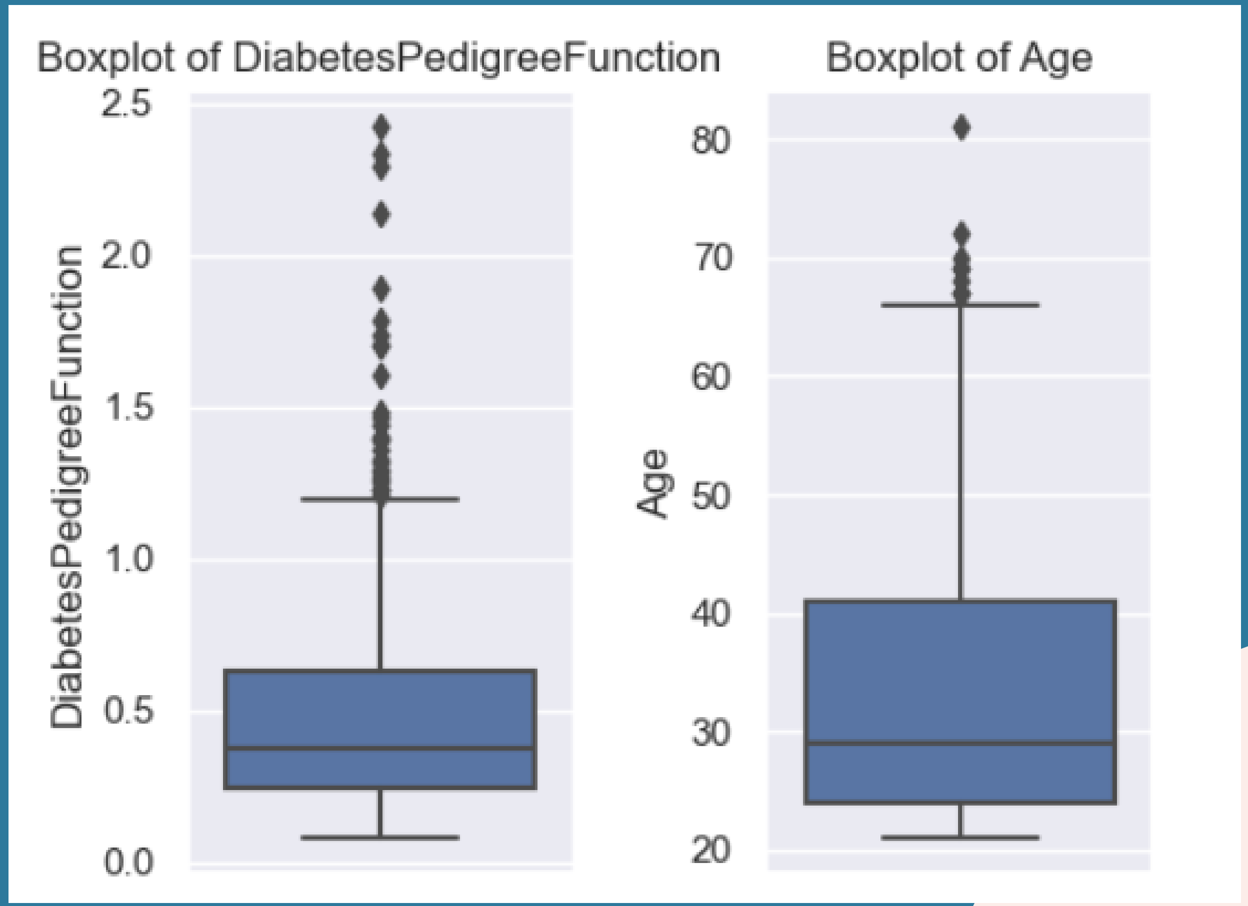








Checking outliers: Glucose is the only feature without any outliers



4.Building models:

1. Logistic regression

```
# One-hot encode categorical variables
diabetes_scaled = pd.get_dummies(diabetes_scaled, columns=['BMI_Category', 'Age_Group'], drop_first=True)

# Logistic Regression

X_lr = diabetes_scaled[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
                        'DiabetesPedigreeFunction', 'Age', 'Glucose_Insulin_Interact',
                        'BMI_Category_Normal Weight', 'BMI_Category_Overweight', 'BMI_Category_Obese I',
                        'BMI_Category_Obese II', 'BMI_Category_Obese III',
                        'Age_Group_30-40', 'Age_Group_40-50', 'Age_Group_50-60', 'Age_Group_60-70', 'Age_Group_70+']]
y_lr = diabetes_scaled[['Outcome']]

X_train_lr, X_test_lr, y_train_lr, y_test_lr = train_test_split(X_lr, y_lr, test_size=0.2, random_state=42)

model_lr = LogisticRegression(penalty="l2", max_iter=100, solver="lbfgs", random_state=42)
model_lr.fit(X_train_lr, y_train_lr.values.ravel())
y_estimated_lr = model_lr.predict(X_test_lr)

print("Logistic Regression:")
print("Accuracy: ", accuracy_score(y_test_lr, y_estimated_lr))
print("\nConfusion Matrix: ", confusion_matrix(y_test_lr, y_estimated_lr))
print("\nClassification report: ", classification_report(y_test_lr, y_estimated_lr))
```

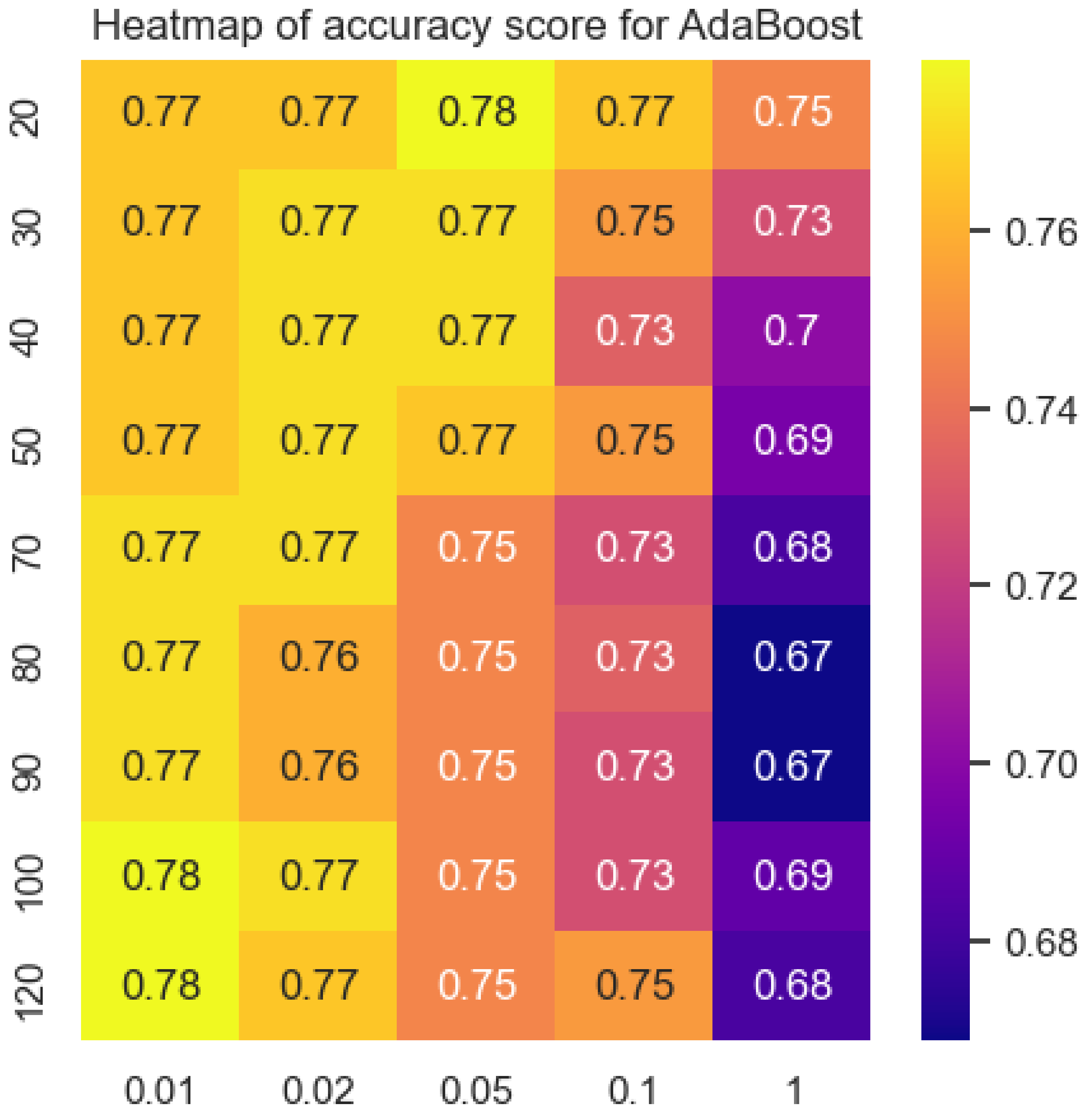
accuracy score: 0.77

2. Random forest

```
n_estimators_rf = [20, 30, 40, 50, 70, 80, 90, 100, 120]
max_depth_rf = np.arange(10, 30)
accuracy_matrix_rf = np.zeros([len(n_estimators_rf), len(max_depth_rf)])

for i, estimator in enumerate(n_estimators_rf):
    for j, depth in enumerate(max_depth_rf):
        rf_model = RandomForestClassifier(n_estimators=estimator, max_depth=depth, random_state=42)
        rf_model.fit(X_train_lr, y_train_lr.values.ravel())
        y_estimated_rf = rf_model.predict(X_test_lr)
        accuracy_matrix_rf[i, j] = accuracy_score(y_test_lr, y_estimated_rf)

best_accuracy_rf = np.max(accuracy_matrix_rf)
best_params_rf = np.unravel_index(np.argmax(accuracy_matrix_rf), accuracy_matrix_rf.shape)
best_n_estimators_rf = n_estimators_rf[best_params_rf[0]]
best_max_depth_rf = max_depth_rf[best_params_rf[1]]
```

Best accuracy score: 0.79

Best accuracy score: 0.78



3. AdaBoost

```python
# AdaBoost Grid Search

base_classifier = DecisionTreeClassifier(max_depth=2)
n_estimators_ab = [20, 30, 40, 50, 70, 80, 90, 100, 120]
learning_rate_ab = [0.01, 0.02, 0.05, 0.1, 1]
accuracy_matrix_ab = np.zeros([len(n_estimators_ab), len(learning_rate_ab)])

for i, estimator in enumerate(n_estimators_ab):
    for j, rate in enumerate(learning_rate_ab):
        adaboost_class = AdaBoostClassifier(base_estimator=base_classifier,
                                            n_estimators=estimator,
                                            learning_rate=rate,
                                            random_state=42)
        adaboost_class.fit(X_train_lr, y_train_lr.values.ravel())
        y_estimated_ab = adaboost_class.predict(X_test_lr)
        accuracy_matrix_ab[i, j] = accuracy_score(y_test_lr, y_estimated_ab)
```

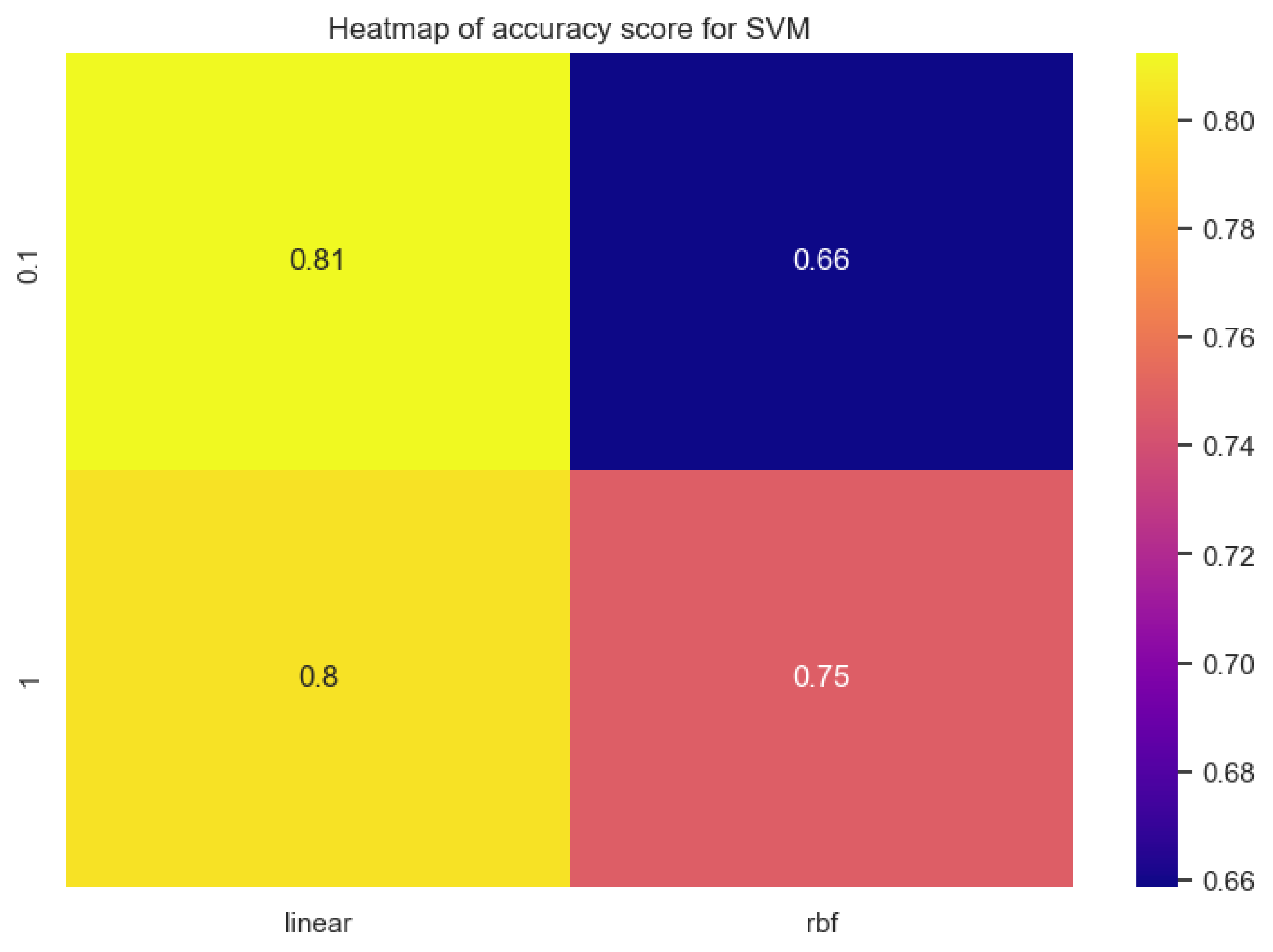
4. LightBGM

```
train_data=lgb.Dataset(X_train_lr,label=y_train_lr)
test_data=lgb.Dataset(X_test_lr,label=y_test_lr,reference=train_data)

params={"objective":"binary",
        "metric":"mse",
        "boosting_type":"gbdt",
        "num_boost_round":100,
        "learning_rate":0.05,
        "max_depth":6,
        "num_leaves":30,
        "feature_fraction":0.9}

# train model
lgb_model = lgb.train(params,train_data,valid_sets=[test_data])

# make predictions
predictions=lgb_model.predict(X_test_lr,num_iteration=lgb_model.best_iteration)
binary_predictions = [1 if pred>=0.5 else 0 for pred in predictions]

# evalutae the model
lbg_accuracy=accuracy_score(y_test_lr,binary_predictions)
print("Accuracy of LightBMG model is ",lbg_accuracy)
```

accuracy score: 0.76

5. SVM

```
Best SVM model parameters:
{'C': 0.1, 'kernel': 'linear'}
The best SVM model accuracy is 0.8123333333333335
```



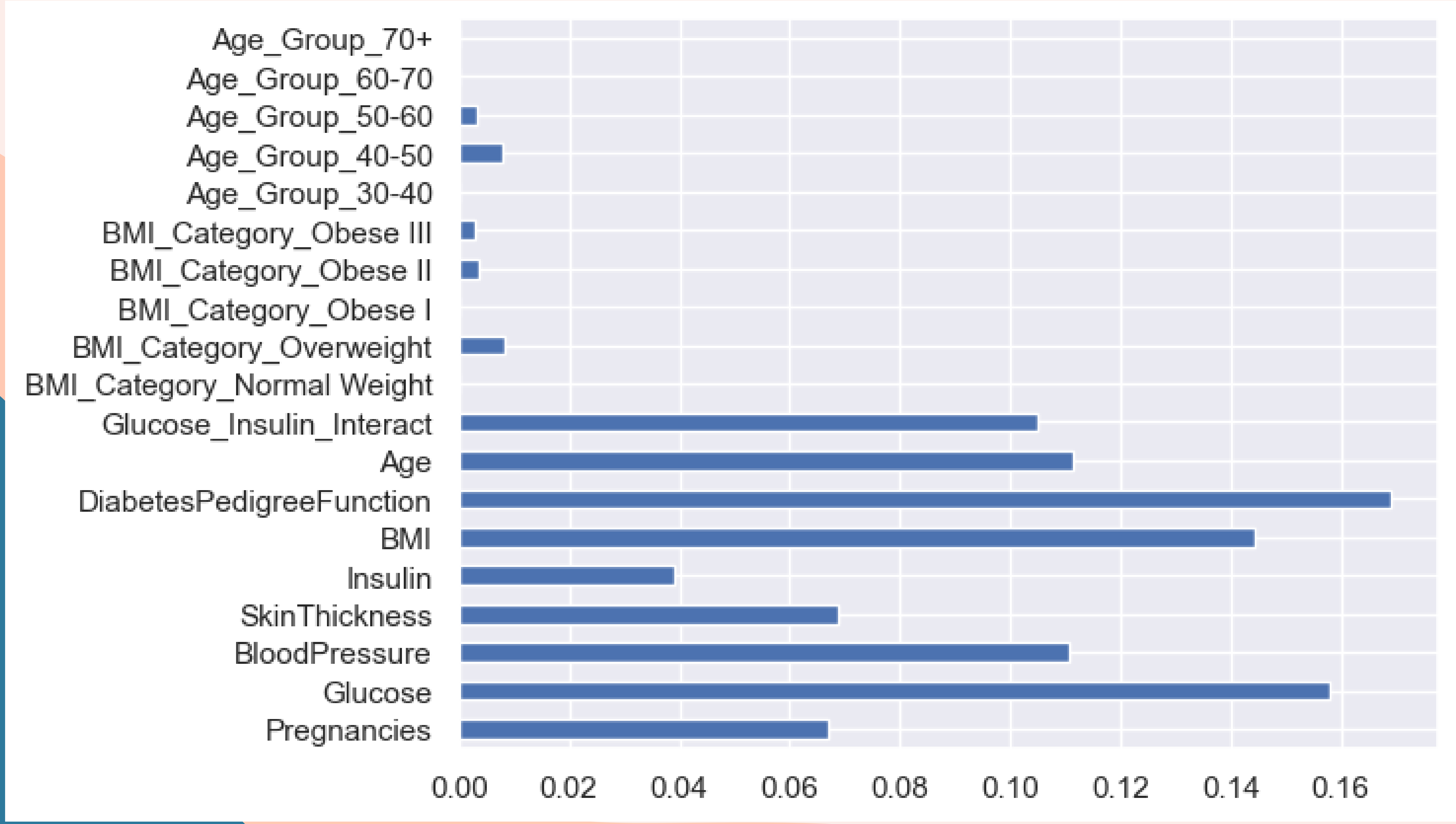
6. XG Boost Classifier

```
Best XGBoost model parameters:
{'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 50, 'subsample': 0.8}
The best XGBoost model accuracy is 0.783379981340797
```

Accuracy Matrix

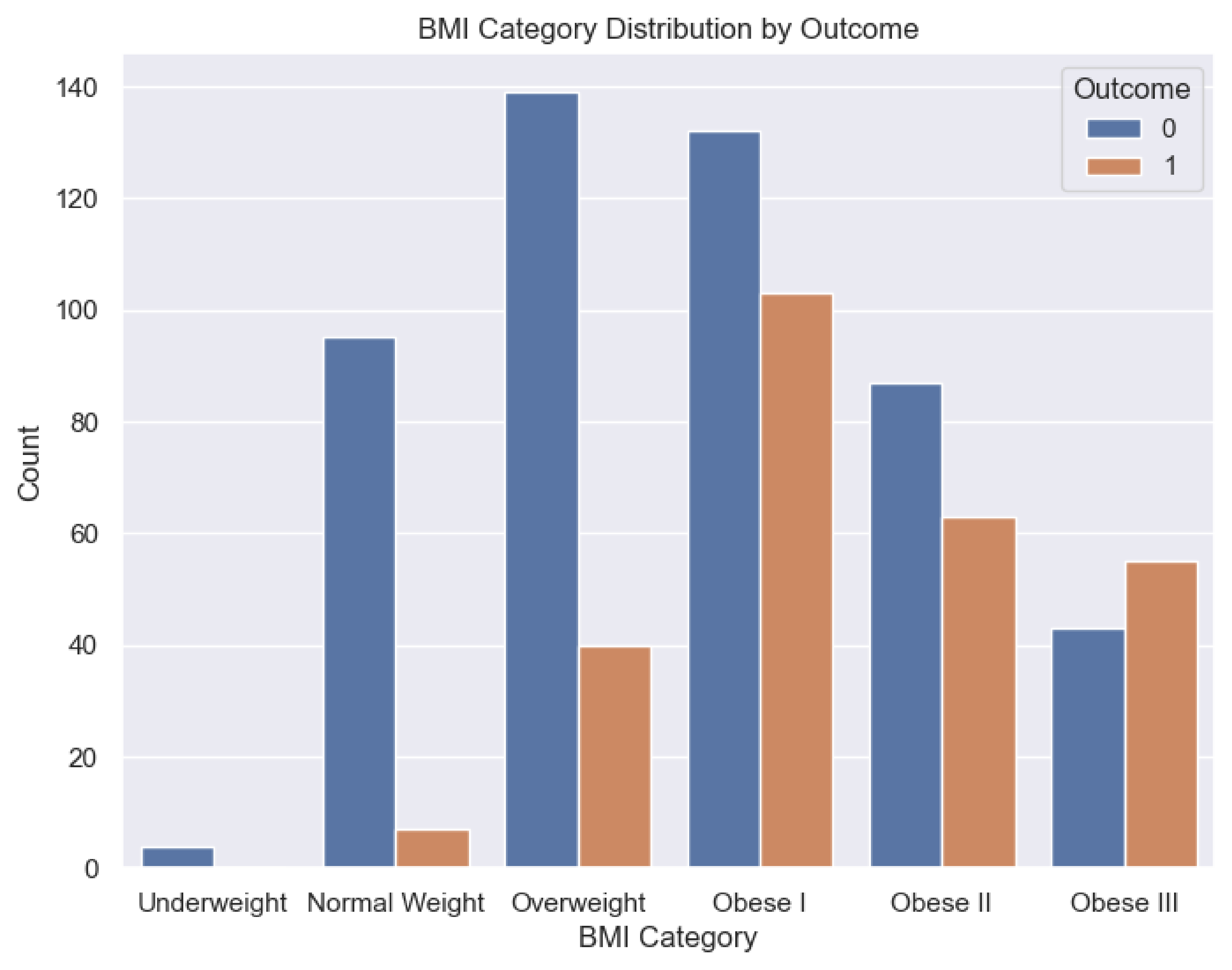
```
[[0.70030654 0.70683727 0.75737705 0.73779821 0.76871918 0.7622551
  0.70686392 0.7100893  0.76060243 0.74435559 0.76222844 0.7605891
  0.71338131 0.7198454  0.75409836 0.73775823 0.76544049 0.74919366
  0.78175397 0.78010129 0.76871918 0.76220179 0.75731041 0.7638278
  0.75731041 0.7703452  0.77361056 0.76378782 0.75893643 0.77031854
  0.75572438 0.75888311 0.75732374 0.76867919 0.75408503 0.76705318]
 [0.75893643 0.76707983 0.76545382 0.76871918 0.76218846 0.75565774
  0.76222844 0.75729708 0.75896308 0.76381447 0.75241903 0.74260962
  0.75728375 0.76057577 0.75403172 0.76546715 0.75404505 0.75893643
  0.70848994 0.72311076 0.76222844 0.75087298 0.77359723 0.7654938
  0.7117553  0.71824603 0.75733707 0.74270292 0.76869252 0.75572438
  0.71339464 0.72152472 0.7605891  0.73451953 0.76869252 0.74594162]
 [0.78337998 0.77527656 0.76548047 0.77198454 0.76221511 0.76380115
  0.76709316 0.76709316 0.76057577 0.77031854 0.76545382 0.76705318
  0.7621618  0.75728375 0.76381447 0.75732374 0.76545382 0.76709316
  0.7621618  0.76875916 0.76214847 0.7621618  0.76218846 0.76706651
  0.77029188 0.76871918 0.76381447 0.7638278  0.75563108 0.76221511
  0.7589231  0.7638278  0.75405838 0.76054911 0.7589231  0.75565774]]
```

Feature Importance



Features that make most contributions are DiabetesPedigreeFunction, Glucose, BMI, Age, BloodPressure.

Feature Engineering

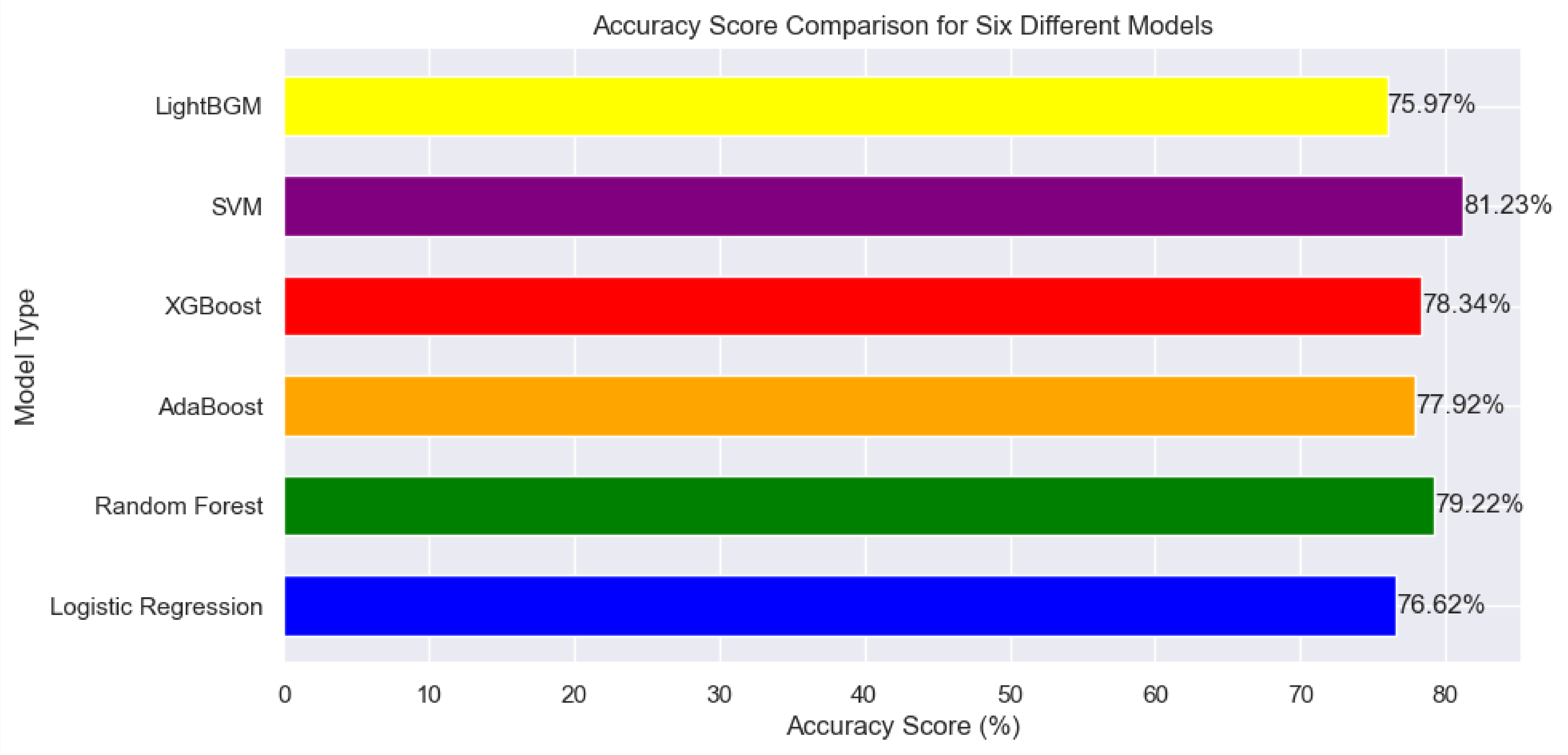


The following new features were added:

- BMI_Cateogy
- Glucose_Insulin_Interact
- Age_Group

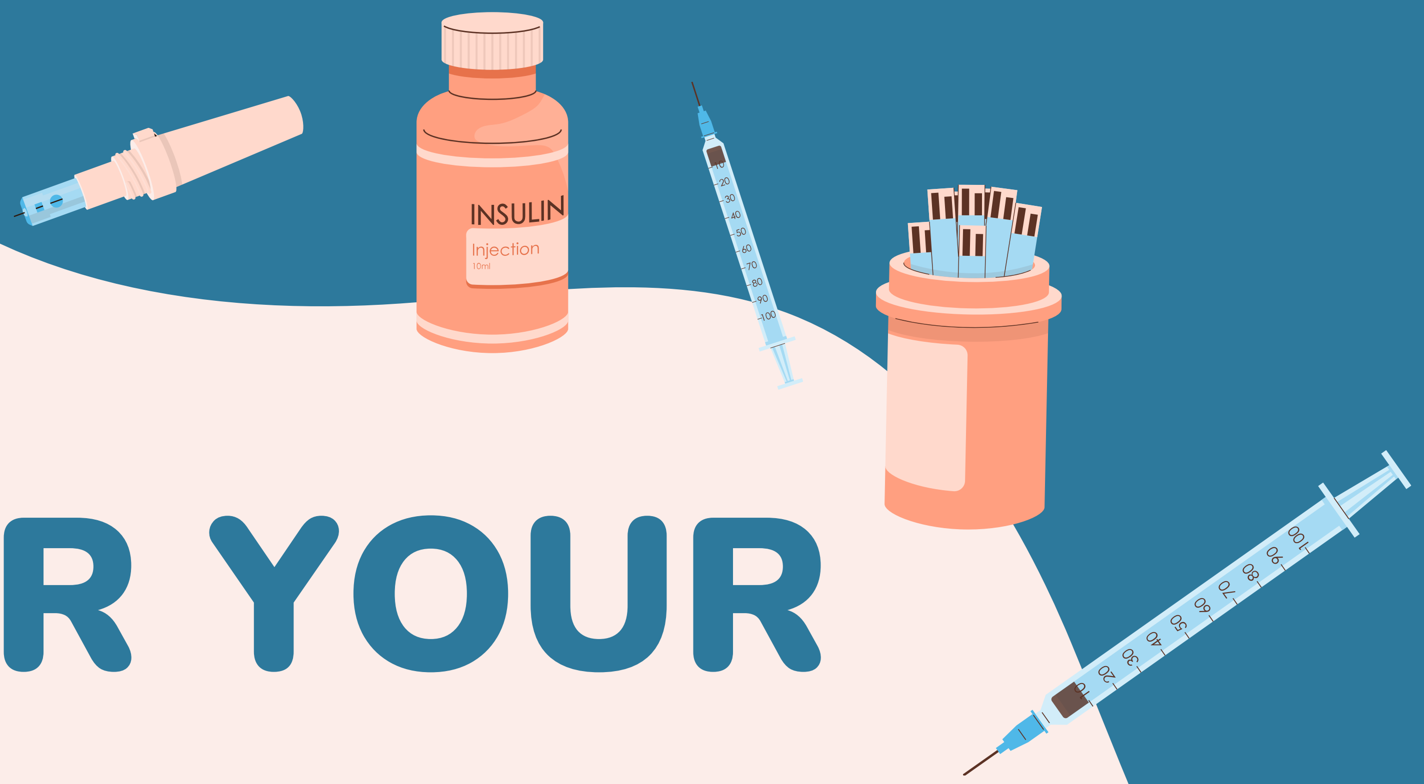
Performance Comparison

And the winner is!!!



Further Improvements

- Better Feature Engineering
- Regularization
- Better Scaling of Data



THANK YOU FOR YOUR ATTENTION!
WE ARE NOW OPEN FOR ANY QUESTIONS!