**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

%matplotlib inline

[2]

df = pd.read\_csv("diabetes.csv")

df.head()

[3]

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 768 entries, 0 to 767

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Pregnancies 768 non-null int64

1 Glucose 768 non-null int64

2 BloodPressure 768 non-null int64

3 SkinThickness 768 non-null int64

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

[4]

df.isnull().sum()

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

[5]

pd.set\_option('display.float\_format', '{**:.2f**}'.format)

[6]

df.describe()

[7]

df.columns

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],

dtype='object')

[9]

feature\_columns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

**for** column **in** feature\_columns:

print("============================================")

print(**f**"{column} ==> Missing zeros : {len(df.loc[df[column] == 0])}") ============================================

Glucose ==> Missing zeros : 5

============================================

BloodPressure ==> Missing zeros : 35

============================================

SkinThickness ==> Missing zeros : 227

============================================

Insulin ==> Missing zeros : 374

============================================

BMI ==> Missing zeros : 11

============================================

DiabetesPedigreeFunction ==> Missing zeros : 0

============================================

Age ==> Missing zeros : 0

[10]

**from** sklearn.impute **import** SimpleImputer

fill\_values = SimpleImputer(missing\_values=0, strategy="mean", copy=False)

df[feature\_columns] = fill\_values.fit\_transform(df[feature\_columns]) [11]

plt.figure(figsize=(20, 20))

**for** i, column **in** enumerate(df.columns, 1):

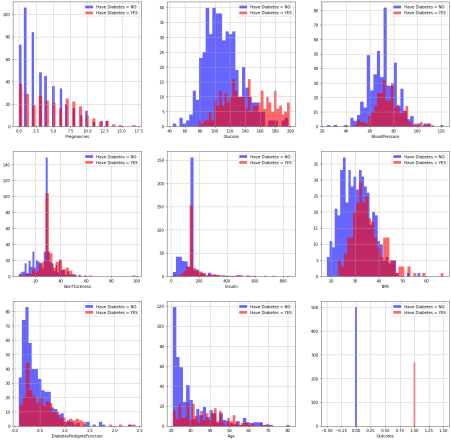
plt.subplot(3, 3, i)

df[df["Outcome"] == 0][column].hist(bins=35, color='blue', label='Have Diabete s = NO', alpha=0.6)

df[df["Outcome"] == 1][column].hist(bins=35, color='red', label='Have Diabetes = YES', alpha=0.6)

plt.legend()

plt.xlabel(column)

[12]

**from** sklearn.model\_selection **import** train\_test\_split

in\_col = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', ' BMI', 'DiabetesPedigreeFunction', 'Age']

X = df[in\_col]

y = df.Outcome

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_st ate=42)

[13]

**from** sklearn.metrics **import** confusion\_matrix, accuracy\_score, classification\_repot **def** evaluate(model, X\_train, X\_test, y\_train, y\_test):

y\_test\_pred = model.predict(X\_test)

y\_train\_pred = model.predict(X\_train)

print("TRAINING RESULTS: \n")

clf\_report = pd.DataFrame(classification\_report(y\_train, y\_train\_pred, output\_ dict=True))

print(**f**"CONFUSION MATRIX:\n{confusion\_matrix(y\_train, y\_train\_pred)}") print(**f**"ACCURACY SCORE:\n{accuracy\_score(y\_train, y\_train\_pred)**:.4f**}")

print(**f**"CLASSIFICATION REPORT:\n{clf\_report}")

print("TESTING RESULTS: \n")

clf\_report = pd.DataFrame(classification\_report(y\_test, y\_test\_pred, output\_d ct=True))

print(**f**"CONFUSION MATRIX:\n{confusion\_matrix(y\_test, y\_test\_pred)}") print(**f**"ACCURACY SCORE:\n{accuracy\_score(y\_test, y\_test\_pred)**:.4f**}") print(**f**"CLASSIFICATION REPORT:\n{clf\_report}")

[14]

*#Boosting*

**from** sklearn.ensemble **import** AdaBoostClassifier

ada\_boost\_clf = AdaBoostClassifier(n\_estimators=30)

ada\_boost\_clf.fit(X\_train, y\_train)

evaluate(ada\_boost\_clf, X\_train, X\_test, y\_train, y\_test)

TRAINING RESULTS:

CONFUSION MATRIX:

[[310 39]

[ 51 137]]

ACCURACY SCORE:

0.8324

CLASSIFICATION REPORT:

0 1 accuracy macro avg weighted avg

precision 0.86 0.78 0.83 0.82 0.83

recall 0.89 0.73 0.83 0.81 0.83

f1-score 0.87 0.75 0.83 0.81 0.83

support 349.00 188.00 0.83 537.00 537.00

TESTING RESULTS:

CONFUSION MATRIX:

[[123 28]

[ 27 53]]

ACCURACY SCORE:

0.7619

CLASSIFICATION REPORT:

0 1 accuracy macro avg weighted avg

precision 0.82 0.65 0.76 0.74 0.76

recall 0.81 0.66 0.76 0.74 0.76

f1-score 0.82 0.66 0.76 0.74 0.76

support 151.00 80.00 0.76 231.00 231.00

[16]

*#Bagging*

**from** sklearn.ensemble **import** BaggingClassifier

bag\_boost\_clf = BaggingClassifier(n\_estimators=30)

bag\_boost\_clf.fit(X\_train, y\_train)

evaluate(bag\_boost\_clf, X\_train, X\_test, y\_train, y\_test)

TRAINING RESULTS:

CONFUSION MATRIX:

[[349 0]

[ 0 188]]

ACCURACY SCORE:

1.0000

CLASSIFICATION REPORT:

0 1 accuracy macro avg weighted avg

precision 1.00 1.00 1.00 1.00 1.00

recall 1.00 1.00 1.00 1.00 1.00

f1-score 1.00 1.00 1.00 1.00 1.00

support 349.00 188.00 1.00 537.00 537.00

TESTING RESULTS:

CONFUSION MATRIX:

[[119 32]

[ 26 54]]

ACCURACY SCORE:

0.7489

CLASSIFICATION REPORT:

0 1 accuracy macro avg weighted avg

precision 0.82 0.63 0.75 0.72 0.75

recall 0.79 0.68 0.75 0.73 0.75

f1-score 0.80 0.65 0.75 0.73 0.75

support 151.00 80.00 0.75 231.00 231.00

[17]

scores ={

'Ensemble tech': {

'AdaBoostTest': accuracy\_score(y\_test, ada\_boost\_clf.predict(X\_test)), 'BaggingTest' : accuracy\_score(y\_test, bag\_boost\_clf.predict(X\_test)), }

}

[18]

scores\_df = pd.DataFrame(scores)

scores\_df.plot(kind='barh', figsize=(15, 8))

<AxesSubplot:>

