HEALTH CARE

CAPSTONE PROJECT

tableau-
https://public.tableau.com/app/profile/aarthy8395/viz/Healthcare_capstone_Aarthy/Dashboard1?publish=yes

Build a model to accurately predict whether the patients in the dataset have diabetes or not.

Project Task: Week 1

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

```
df.describe()
```

replace with mean and median

```
df['Glucose']=df['Glucose'].replace(0,df['Glucose'].median())

df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].median())

df['SkinThickness']=df['SkinThickness'].replace(0,df['SkinThickness'].mean())

df['Insulin']=df['Insulin'].replace(0,df['Insulin'].median())

df['BMI']=df['BMI'].replace(0,df['BMI'].mean())
```

.2. Visually explore these variables using histograms. Treat the missing values accordingly.

```
df.isnull().any()

df.hist(figsize=(15,15))
```

3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

```
df_count.plot(kind='bar')
```

Project Task: Week 2

Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.

```
import seaborn as sns

sns.countplot(df['Outcome'])
```

imbalance so oversampling-smote

```
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state = 2)

x_train_res, y_train_res = sm.fit_resample(x_train, y_train.ravel())

print("After smote '1' {}".format(sum(y_train_res == 1)))

print("After smote'0': {}".format(sum(y_train_res == 0)))
```

2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

```
sns.pairplot(df)
```

3. Perform correlation analysis. Visually explore it using a heat map.

```
plt.figure(figsize=(15,7))

sns.heatmap(df.corr(),annot=True)
```

Project Task: Week 3

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.

2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

```
from sklearn.linear_model import LogisticRegression # logistic reg is better binary classification

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score

lr=LogisticRegression()

lr.fit(x_train,y_train)

score=lr.score(x_train,y_train)

y_pred_lr=lr.predict(x_test)

acc=accuracy_score(y_test,y_pred_lr)

print('acc is :',acc *100)
```

knn

```
from sklearn.neighbors import KNeighborsClassifier

knn= KNeighborsClassifier(n_neighbors=6)
```

```
knn.fit(x_train, y_train)

score=knn.score(x_train,y_train)

y_pred_knn=knn.predict(x_test)

acc=accuracy_score(y_test,y_pred_knn)

        print('acc is :',acc *100)

from sklearn.model_selection import KFold

from sklearn.model_selection import cross_val_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.naive_bayes import GaussianNB

        from sklearn.svm import SVC

for name, model in classification_models:

  kfold = KFold(n_splits=10, random_state=(7), shuffle=(True))

  result = cross_val_score(model, x, y, cv=kfold, scoring='accuracy')

          print("%s: Mean Accuracy = %.2f%% - SD Accuracy = %.2f%%" % (name, result.mean()*100, result.std()*100))

        print(confusion_matrix(y_test,y_pred_lr))

 print(classification_report(y_test,y_pred_lr))

roc_auc_score(y_test,y_pred_lr)
```