In class activity 4

Classification

INDEX

Contents

1. Explanation of data preparation process:

Objective behind taking this topic:

The Behavioral Risk Factor Surveillance System (BRFSS) is a health-related telephone survey that is collected annually by the CDC. Each year, the survey collects responses from over 400,000 Americans on health-related risk behaviors, chronic health conditions, and the use of preventative services. It has been conducted every year since 1984. For this project, a csv of the dataset available on Kaggle for the year 2015 was used. This original dataset contains responses from 441,455 individuals and has 330 features. These features are either questions directly asked of participants, or calculated variables based on individual participant responses.

```
In [3]: df.head()
```

Out[3]:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	5.0
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.0	...	0.0	1.0	3.0
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.0	...	1.0	1.0	5.0
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0

5 rows × 22 columns

Here, we have 22 columns in total where every feature has their own important.

```
In [5]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Diabetes_binary       253680 non-null  float64
 1   HighBP                253680 non-null  float64
 2   HighChol              253680 non-null  float64
 3   CholCheck             253680 non-null  float64
 4   BMI                   253680 non-null  float64
 5   Smoker                253680 non-null  float64
 6   Stroke                253680 non-null  float64
 7   HeartDiseaseorAttack  253680 non-null  float64
 8   PhysActivity          253680 non-null  float64
 9   Fruits                253680 non-null  float64
 10  Veggies               253680 non-null  float64
 11  HvyAlcoholConsump     253680 non-null  float64
 12  AnyHealthcare         253680 non-null  float64
 13  NoDocbcCost           253680 non-null  float64
 14  GenHlth               253680 non-null  float64
 15  MentHlth              253680 non-null  float64
 16  PhysHlth              253680 non-null  float64
 17  DiffWalk              253680 non-null  float64
 18  Sex                   253680 non-null  float64
 19  Age                   253680 non-null  float64
 20  Education             253680 non-null  float64
 21  Income                253680 non-null  float64
dtypes: float64(22)
memory usage: 42.6 MB
```

In preprocessing data getting info is the main part which gives overview about data type.

```
In [6]: df.isnull().sum()
```

```
Out[6]: Diabetes_binary          0
HighBP                   0
HighChol                 0
CholCheck                0
BMI                      0
Smoker                   0
Stroke                   0
HeartDiseaseorAttack     0
PhysActivity             0
Fruits                   0
Veggies                  0
HvyAlcoholConsump        0
AnyHealthcare            0
NoDocbcCost              0
GenHlth                  0
MentHlth                 0
PhysHlth                 0
DiffWalk                 0
Sex                      0
Age                      0
Education                0
Income                   0
dtype: int64
```

Let's check that any null values are here or not. So, we can see that data don't have any NaN values.

```
In [9]: df.shape
```

```
Out[9]: (253680, 22)
```

```
In [10]: # Removing duplicate rows from the dataset
df.drop_duplicates(inplace = True)
```

```
In [11]: df.shape
```

```
Out[11]: (229474, 22)
```

Here three line of code where first code says that we have 253680 rows in data which contains some duplicate data. After removing duplicate dataset we can see that our data became 229474 rows.

Modeling data:

```
In [13]: X = df.drop('Diabetes_binary',axis=1)
y = df['Diabetes_binary']
```

Here we spitted data into train and test where X contain data without Diabete_binary and where y contain only Diabete_binary.

```
In [15]: X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=7)
```

Here 20% data set goes into test size and 80% data goes into train.

2. Classification models:

- ### Model - 1 Random Forest:

```
In [17]: model = RandomForestClassifier()
         model.fit(X_train,y_train)
```

Taking RandomForestClassification function as in model and fit X_train and y_train in that.

```
In [26]: X_100 = X_test.head(100)
```

```
In [28]: prediction = model.predict(X_100)
         print("Predicted class points on Test Set:",prediction)
```

```
Predicted class points on Test Set: [0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.]
```

Here we took Top 100 rows in X_100 to get predication values of X_test.

```
In [19]: y_pred = model.predict(X_test)
         print("------------------------------------------------------------------------------")
         print(f"The accuracy score is: ------>>  {accuracy_score(y_test,y_pred)}")
         print("------------------------------------------------------------------------------")
         print(f"The Confusion Matrix is: ------>> \n{confusion_matrix(y_test,y_pred)}")
         print("------------------------------------------------------------------------------")
         print(f"The Classification Report is: ---->> {classification_report(y_test,y_pred)}")
```

```
------------------------------------------------------------------------
The accuracy score is: ------>>  0.8442313977557468
------------------------------------------------------------------------
The Confusion Matrix is: ------>>
[[37516  1350]
 [ 5799  1230]]
------------------------------------------------------------------------
The Classification Report is: ---->>               precision    recall  f1-score   support

         0.0       0.87      0.97      0.91     38866
         1.0       0.48      0.17      0.26      7029

    accuracy                           0.84     45895
   macro avg       0.67      0.57      0.58     45895
weighted avg       0.81      0.84      0.81     45895
```

Now y_pred gives as predicated values of X_test, Which represented Accuracy, confusion Matrix and Classification report.

- ## Model - 2 Logistic Regressions:

It's seems that this one is Regression model but as per python libraries we are using Logistic Regression in classification.

```
In [31]: model = LogisticRegression()
         model.fit(X_train,y_train)
```

```
C:\Users\win\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (s
tatus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Out[31]: LogisticRegression()
```

Taking LogisticRegression function as in model and fit X_train and y_train in that.

```
In [33]: X_100 = X_test.head(100)
         prediction = model.predict(X_100)
         print("Predicted class points on Test Set:",prediction)
```

```
Predicted class points on Test Set: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0.
 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0.]
```

Here we took Top 100 rows in X_100 to get predication values of X_test.

```
In [34]: y_pred = model.predict(X_test)
         print("-------------------------------------------------------------------------")
         print(f"The accuracy score is: ------>>  {accuracy_score(y_test,y_pred)}")
         print("-------------------------------------------------------------------------")
         print(f"The Confusion Matrix is: ------>> \n{confusion_matrix(y_test,y_pred)}")
         print("-------------------------------------------------------------------------")
         print(f"The Classification Report is: ---->> {classification_report(y_test,y_pred)}")
```

```
-------------------------------------------------------------------------
The accuracy score is: ------>>  0.8503322802048153
-------------------------------------------------------------------------
The Confusion Matrix is: ------>>
[[37971   895]
 [ 5974  1055]]
-------------------------------------------------------------------------
The Classification Report is: ---->>              precision    recall  f1-score   support

         0.0       0.86      0.98      0.92     38866
         1.0       0.54      0.15      0.23      7029

    accuracy                           0.85     45895
   macro avg       0.70      0.56      0.58     45895
weighted avg       0.81      0.85      0.81     45895
```

Now y_pred gives as predicated values of X_test, Which represented Accuracy, confusion Matrix and Classification report.

- **Model - 3 KNN:**

```
In [36]: model = KNeighborsClassifier()
         model.fit(X_train,y_train)

Out[36]: KNeighborsClassifier()
```

Taking KNeighborsClassifier function as in model and fit X_train and y_train in that.

```
In [37]: X_100 = X_test.head(100)
         prediction = model.predict(X_100)
         print("Predicted class points on Test Set:",prediction)

Predicted class points on Test Set: [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0.]

C:\Users\win\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functi
ons (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be e
liminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Here we took Top 100 rows in X_100 to get predication values of X_test.

```
In [38]: y_pred = model.predict(X_test)
         print("-------------------------------------------------------------------------")
         print(f"The accuraccy score is: ------>>  {accuracy_score(y_test,y_pred)}")
         print("-------------------------------------------------------------------------")
         print(f"The Confusion Matrix is: ------>> \n{confusion_matrix(y_test,y_pred)}")
         print("-------------------------------------------------------------------------")
         print(f"The Classification Report is: ---->> {classification_report(y_test,y_pred)}")

C:\Users\win\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarnin
ons (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it a
behavior will change: the default value of `keepdims` will become False, the `axis` over which
liminated, and the value None will no longer be accepted. Set `keepdims` to True or False to av
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)

-------------------------------------------------------------------------
The accuraccy score is: ------>>  0.8328793986273014
-------------------------------------------------------------------------
The Confusion Matrix is: ------>>
[[36821  2045]
 [ 5625  1404]]
-------------------------------------------------------------------------
The Classification Report is: ---->>               precision    recall  f1-score   support

           0.0       0.87      0.95      0.91     38866
           1.0       0.41      0.20      0.27      7029

      accuracy                           0.83     45895
     macro avg       0.64      0.57      0.59     45895
  weighted avg       0.80      0.83      0.81     45895
```

Now y_pred gives as predicated values of X_test, Which represented Accuracy, confusion Matrix and Classification report.

3. Conclusion:

Here we can see that Logistic Regression works best in this case!! Where other remain 2 not perform as well as Logistic Regression perform.

4. Reference:

https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset/