Pima Diabetes

February 6, 2019

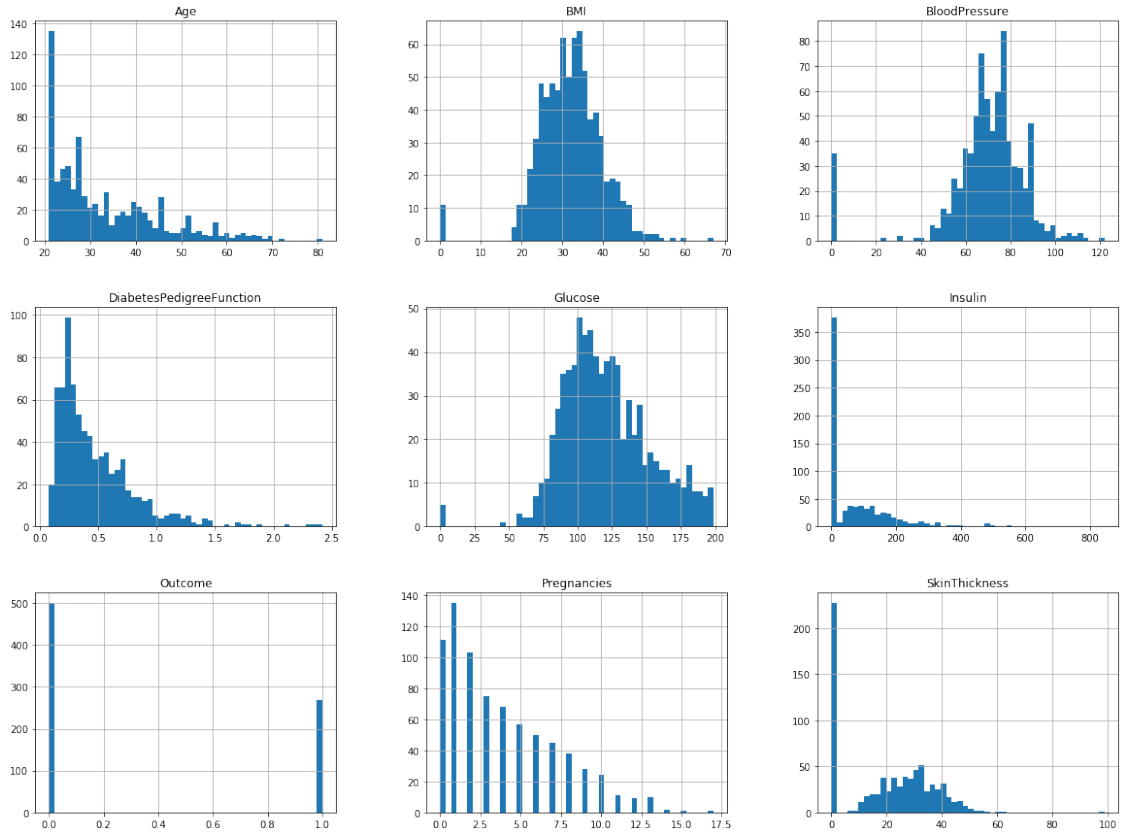
In [1]:
```
#Importing Libraries

        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import scipy.stats as sp
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import accuracy_score as acc
        from mlxtend.feature_selection import SequentialFeatureSelector as sfs
        from sklearn.preprocessing import StandardScaler as ss
        from sklearn.model_selection import cross_val_score as cvs, train_test_split
```

In [2]:
```
#Reading Dataset
        df=pd.read_csv("C:\\Users\\vj0805\\Desktop\\diabetes.csv")
```

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

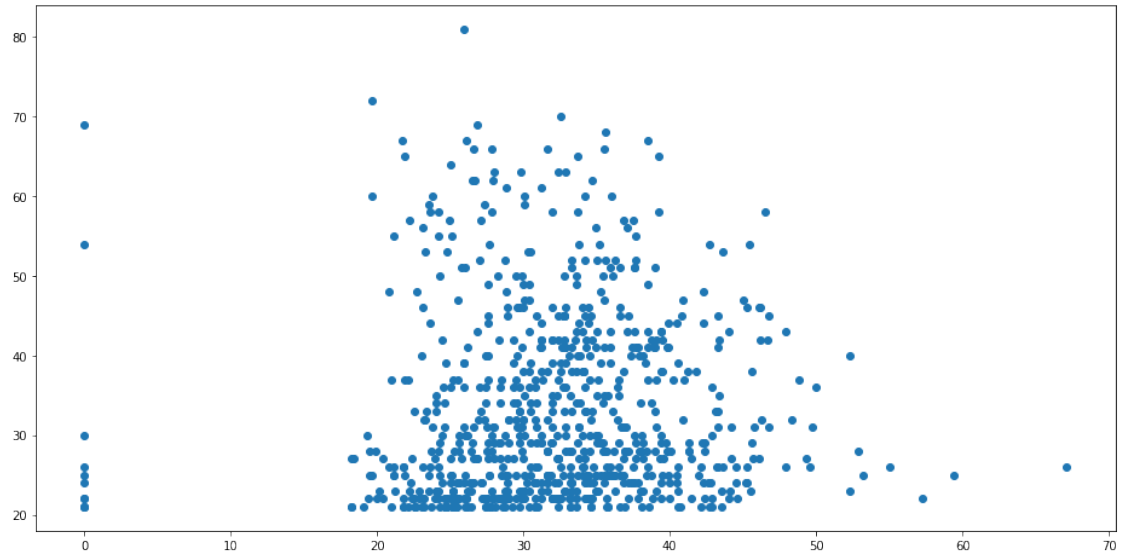
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:
```
# Frequency distribution of various attributes
        df.hist(bins=50, figsize=(20, 15))
        plt.show()
```



In [5]:
```
#Scatter plot to detect outliers
fig,ax=plt.subplots(figsize=(16,8))
ax.scatter(df["BMI"],df["Age"])
```

Out[5]: <matplotlib.collections.PathCollection at 0x1637f07dc18>



```
In [10]: # Z score is calculated and threshold of [-3, 3] is used as 99.7% data is between 3 s
         z=sp.zscore(df)
         threshold=3
         k=np.where(z>threshold)
         k
```

```
Out[10]: (array([  4,   8,  13,  45,  58,  88, 111, 123, 153, 159, 177, 186, 220,
                 228, 228, 247, 286, 298, 330, 370, 370, 371, 395, 409, 415, 445,
                 445, 453, 455, 459, 486, 579, 584, 593, 621, 645, 655, 666, 673,
                 684, 695, 753], dtype=int64),
          array([6, 4, 4, 6, 6, 0, 4, 7, 4, 0, 5, 4, 4, 4, 6, 4, 4, 0, 6, 4, 6, 6,
                 6, 4, 4, 5, 6, 7, 0, 7, 4, 3, 4, 6, 6, 4, 4, 7, 5, 7, 4, 4],
                dtype=int64))
```

```
In [11]: #IQR is calculated to remove outliers
         Q1 = df.quantile(0.25)
         Q3 = df.quantile(0.75)
         IQR = Q3 - Q1
         IQR
```

```
Out[11]: Pregnancies                 5.0000
         Glucose                    41.2500
         BloodPressure              18.0000
         SkinThickness              32.0000
         Insulin                   127.2500
         BMI                         9.3000
         DiabetesPedigreeFunction    0.3825
         Age                        17.0000
         Outcome                     1.0000
         dtype: float64
```

```
In [12]: (df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))
```

```
Out[12]:      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
         0          False    False          False          False    False  False
         1          False    False          False          False    False  False
         2          False    False          False          False    False  False
         3          False    False          False          False    False  False
         4          False    False          False          False    False  False
         5          False    False          False          False    False  False
         6          False    False          False          False    False  False
         7          False    False           True          False    False  False
         8          False    False          False          False     True  False
         9          False    False          False          False    False   True
         10         False    False          False          False    False  False
         11         False    False          False          False    False  False
```

12	False	False	False	False	False	False
13	False	False	False	False	True	False
14	False	False	False	False	False	False
15	False	False	True	False	False	False
16	False	False	False	False	False	False
17	False	False	False	False	False	False
18	False	False	True	False	False	False
19	False	False	False	False	False	False
20	False	False	False	False	False	False
21	False	False	False	False	False	False
22	False	False	False	False	False	False
23	False	False	False	False	False	False
24	False	False	False	False	False	False
25	False	False	False	False	False	False
26	False	False	False	False	False	False
27	False	False	False	False	False	False
28	False	False	False	False	False	False
29	False	False	False	False	False	False
..	...	...	...	...	...	...
738	False	False	False	False	False	False
739	False	False	False	False	False	False
740	False	False	False	False	False	False
741	False	False	False	False	False	False
742	False	False	False	False	False	False
743	False	False	False	False	False	False
744	False	False	False	False	False	False
745	False	False	False	False	False	False
746	False	False	False	False	False	False
747	False	False	False	False	False	False
748	False	False	False	False	False	False
749	False	False	False	False	False	False
750	False	False	False	False	False	False
751	False	False	False	False	False	False
752	False	False	False	False	False	False
753	False	False	False	False	True	False
754	False	False	False	False	False	False
755	False	False	False	False	False	False
756	False	False	False	False	False	False
757	False	False	False	False	False	False
758	False	False	False	False	False	False
759	False	False	False	False	False	False
760	False	False	False	False	False	False
761	False	False	False	False	False	False
762	False	False	False	False	False	False
763	False	False	False	False	False	False
764	False	False	False	False	False	False
765	False	False	False	False	False	False
766	False	False	False	False	False	False

```
767        False    False          False          False    False  False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                        True  False    False
5                       False  False    False
6                       False  False    False
7                       False  False    False
8                       False  False    False
9                       False  False    False
10                      False  False    False
11                      False  False    False
12                       True  False    False
13                      False  False    False
14                      False  False    False
15                      False  False    False
16                      False  False    False
17                      False  False    False
18                      False  False    False
19                      False  False    False
20                      False  False    False
21                      False  False    False
22                      False  False    False
23                      False  False    False
24                      False  False    False
25                      False  False    False
26                      False  False    False
27                      False  False    False
28                      False  False    False
29                      False  False    False
..                        ...    ...      ...
738                     False  False    False
739                     False  False    False
740                     False  False    False
741                     False  False    False
742                     False  False    False
743                     False  False    False
744                     False  False    False
745                     False  False    False
746                     False  False    False
747                     False  False    False
748                     False  False    False
749                     False  False    False
750                     False  False    False
751                     False  False    False
```

```
752                     False  False    False
753                     False  False    False
754                     False  False    False
755                     False  False    False
756                     False  False    False
757                     False  False    False
758                     False  False    False
759                     False  False    False
760                     False  False    False
761                     False  False    False
762                     False  False    False
763                     False  False    False
764                     False  False    False
765                     False  False    False
766                     False  False    False
767                     False  False    False

[768 rows x 9 columns]
```

In [13]:
```
#Outliers are removed
df_out=df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df_out.shape
```

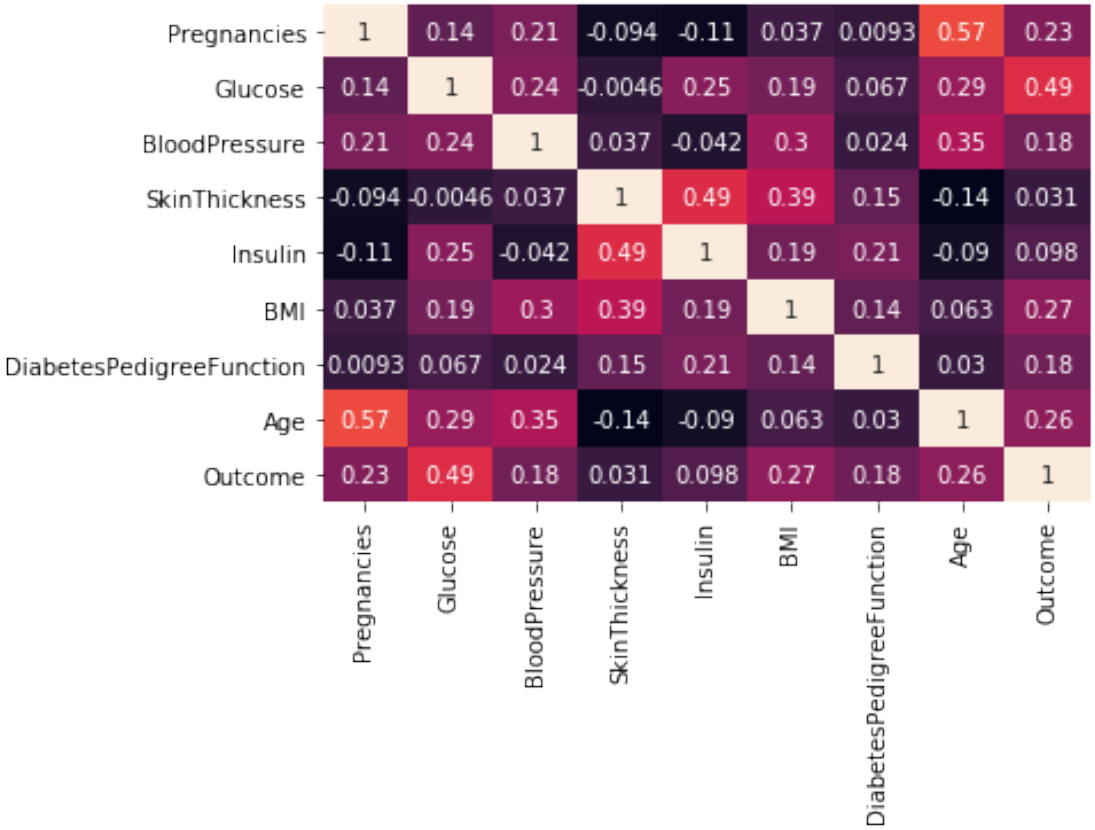
Out[13]:
```
(639, 9)
```

In [14]:
```
df_out.head()
```

Out[14]:

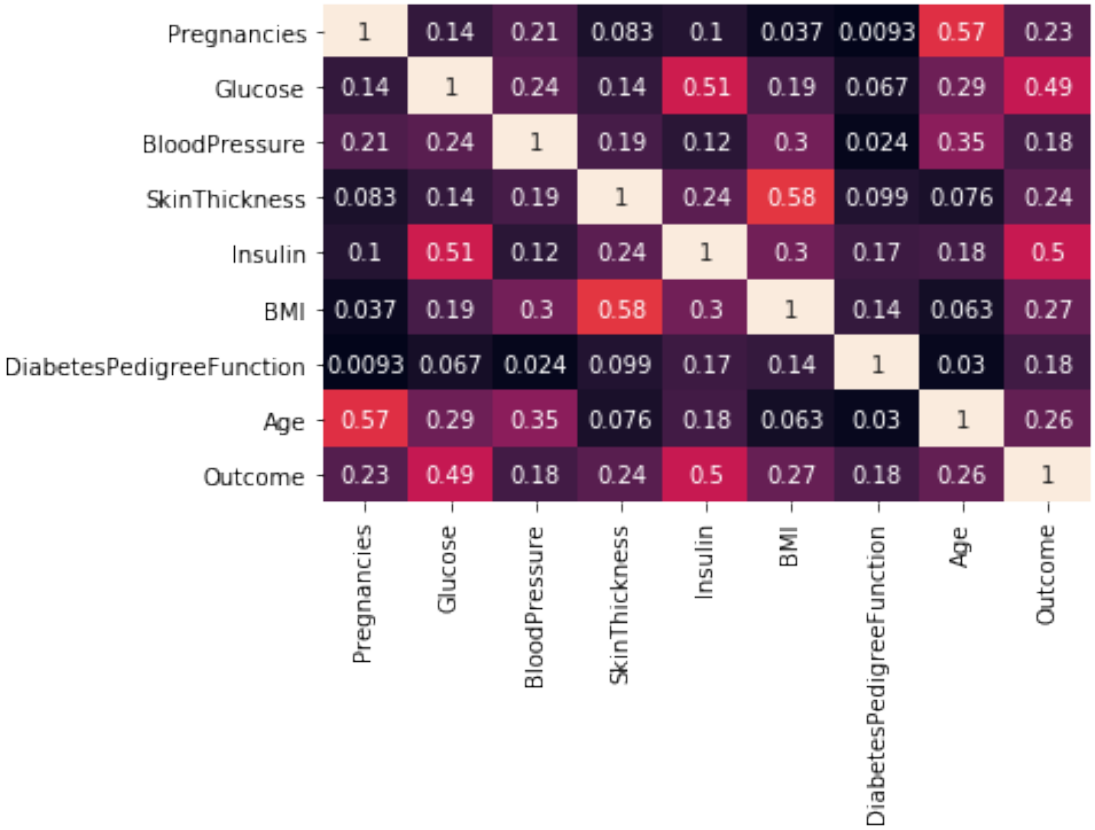
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
5	5	116	74	0	0	25.6	0.201	30	0

In [15]:
```
#correlation map is plotted to see the correlation between attributes
corr=df_out.corr()
sns.heatmap(corr,annot=True,cbar=False)
df=df_out
```




```
In [16]: #Replacing all 0 values by NaN to do imputation
         df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','Blood
         data_ins = df[df['Insulin'].notnull()]

In [17]: # As input and output is correlated so instead of doing mean,median imputation
         #we should do the imputation based on the dependent variable i.e. output
         data_ins = data_ins[['Insulin', 'Outcome']].groupby(['Outcome'])[['Insulin']].median()
         data_ins

Out[17]:    Outcome  Insulin
         0        0     97.5
         1        1    157.5

In [18]: df.loc[(df['Outcome'] == 0 ) & (df['Insulin'].isnull()), 'Insulin'] = 97.5
         df.loc[(df['Outcome'] == 1 ) & (df['Insulin'].isnull()), 'Insulin'] = 157.5

In [19]: data_gl = df[df['Glucose'].notnull()]
         data_gl = data_gl[['Glucose', 'Outcome']].groupby(['Outcome'])[['Glucose']].median().r
         data_gl

Out[19]:    Outcome  Glucose
         0        0      107
         1        1      137
```

```
In [20]: df.loc[(df['Outcome'] == 0 ) & (df['Glucose'].isnull()), 'Glucose'] = 107
         df.loc[(df['Outcome'] == 1 ) & (df['Glucose'].isnull()), 'Glucose'] = 137

In [21]: data_bp = df[df['BloodPressure'].notnull()]
         data_bp = data_bp[['BloodPressure', 'Outcome']].groupby(['Outcome'])[['BloodPressure']
         data_bp

Out[21]:    Outcome  BloodPressure
         0        0             70
         1        1             74

In [22]: df.loc[(df['Outcome'] == 0 ) & (df['BloodPressure'].isnull()), 'BloodPressure'] = 70
         df.loc[(df['Outcome'] == 1 ) & (df['BloodPressure'].isnull()), 'BloodPressure'] = 74

In [23]: data_bmi = df[df['BMI'].notnull()]
         data_bmi = data_bmi[['BMI', 'Outcome']].groupby(['Outcome'])[['BMI']].median().reset_
         data_bmi

Out[23]:    Outcome   BMI
         0        0  30.4
         1        1  33.9

In [24]: df.loc[(df['Outcome'] == 0 ) & (df['BMI'].isnull()), 'BMI'] = 30.1
         df.loc[(df['Outcome'] == 1 ) & (df['BMI'].isnull()), 'BMI'] = 34.3

In [25]: data_sk = df[df['SkinThickness'].notnull()]
         data_sk = data_sk[['SkinThickness', 'Outcome']].groupby(['Outcome'])[['SkinThickness']
         data_sk

Out[25]:    Outcome  SkinThickness
         0        0           27.0
         1        1           32.0

In [26]: df.loc[(df['Outcome'] == 0 ) & (df['SkinThickness'].isnull()), 'SkinThickness'] = 22
         df.loc[(df['Outcome'] == 1 ) & (df['SkinThickness'].isnull()), 'SkinThickness'] = 27

In [27]: cor=df.corr()
         sns.heatmap(cor,annot=True,cbar=False)

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1637f9d0b38>
```




```
In [30]: X_train, X_test, y_train, y_test = train_test_split(df.values[:,:-1],df.values[:,-1:]
         y_train = y_train.ravel()
         y_test = y_test.ravel()
         print('Training dataset shape:', X_train.shape, y_train.shape)
         print('Testing dataset shape:', X_test.shape, y_test.shape)

         clf = RandomForestClassifier(n_estimators=100, n_jobs=-1)

         # Build step forward feature selection
         sfs1 = sfs(clf,
                    k_features=6,
                    forward=True,
                    floating=False,
                    verbose=2,
                    scoring='accuracy',
                    cv=5)

         # Perform SFFS
         sfs1 = sfs1.fit(X_train, y_train)

Training dataset shape: (479, 8) (479,)
```

```
Testing dataset shape: (160, 8) (160,)


[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    4.7s finished

[2019-02-06 19:45:11] Features: 1/6 -- score: 0.8267543859649124[Parallel(n_jobs=1)]: Using ba
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    3.9s finished

[2019-02-06 19:45:15] Features: 2/6 -- score: 0.8600877192982456[Parallel(n_jobs=1)]: Using ba
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    3.4s finished

[2019-02-06 19:45:18] Features: 3/6 -- score: 0.8767982456140352[Parallel(n_jobs=1)]: Using ba
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    3.0s finished

[2019-02-06 19:45:21] Features: 4/6 -- score: 0.8935526315789474[Parallel(n_jobs=1)]: Using ba
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.6s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    2.6s finished

[2019-02-06 19:45:24] Features: 5/6 -- score: 0.8892982456140353[Parallel(n_jobs=1)]: Using ba
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.6s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    1.8s finished

[2019-02-06 19:45:26] Features: 6/6 -- score: 0.8892543859649124
```

```
In [31]: feat_cols = list(sfs1.k_feature_idx_)
         print(feat_cols)
```

```
[0, 1, 2, 4, 5, 7]
```

```
In [32]: sc_x=ss()
         X_train=sc_x.fit_transform(X_train)
         X_test=sc_x.transform(X_test)
```

```
In [33]: # Testing Model with original features
         clf = RandomForestClassifier(n_estimators=1500, random_state=42, max_depth=3)
         clf.fit(X_train, y_train)

         y_train_pred = clf.predict(X_train)
         print('Training accuracy on all features: %.3f' % acc(y_train, y_train_pred))

         y_test_pred = clf.predict(X_test)
         print('Testing accuracy on all features: %.3f' % acc(y_test, y_test_pred))
```

```
Training accuracy on all features: 0.906
Testing accuracy on all features: 0.894
```

```
In [34]: # Testing Model with selectrd features features
         clf = RandomForestClassifier(n_estimators=1000, random_state=42, max_depth=4)
         clf.fit(X_train[:, feat_cols], y_train)

         y_train_pred = clf.predict(X_train[:, feat_cols])
         print('Training accuracy on selected features: %.3f' % acc(y_train, y_train_pred))

         y_test_pred = clf.predict(X_test[:, feat_cols])
         print('Testing accuracy on selected features: %.3f' % acc(y_test, y_test_pred))
```

```
Training accuracy on selected features: 0.923
Testing accuracy on selected features: 0.912
```