```
In [1]: import pandas as pd
        import scipy
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [4]: #load the dataset
        df=pd.read_csv('heartdisease.csv')
```

```
In [3]: df.head(10)
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	1
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
7	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
9	54	1	0	122	286	0	0	116	1	3.2	1	2	2	0

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
In [5]: #checking the null values
        df.isnull().sum()

Out[5]: age         0
        sex         0
        cp          0
        trestbps    0
        chol        0
        fbs         0
        restecg     0
        thalach     0
        exang       0
        oldpeak     0
        slope       0
        ca          0
        thal        0
        target      0
        dtype: int64
```

```
In [6]: #Duplicate Values
        df_duplicate =df.duplicated().any()
        df_duplicate

Out[6]: True

In [7]: df=df.drop_duplicates()

In [8]: df_duplicate =df.duplicated().any()
        df_duplicate

Out[8]: False

In [9]: #Statistical data
        df.describe()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	54.420530	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	2.314570	0.543046
std	9.047970	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	0.613026	0.498970
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.500000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

```
In [10]: #Data processing
         ctr_val = []
         num_val= []

         for column in df.columns:
             if df[column].nunique() <= 10:
                 ctr_val.append(column)
             else:
                 num_val.append(column)

In [11]: ctr_val  #categorical data

Out[11]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

In [12]: num_val   # numerical data

Out[12]: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

In [13]: #Encoding Categorical data
         ctr_val

Out[13]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

In [14]: df['cp'].unique()

Out[14]: array([0, 1, 2, 3], dtype=int64)

In [15]: ctr_val.remove('sex')
         ctr_val.remove('target')
         df= pd.get_dummies(df, columns = ctr_val , drop_first= True)

In [16]: df.head()
```

Out[16]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	52	1	125	212	168	1.0	0	False	False	False	...	False	False	True	False	True	False	False	False	False	True
1	53	1	140	203	155	3.1	0	False	False	False	...	True	False	False	False	False	False	False	False	False	True
2	70	1	145	174	125	2.6	0	False	False	False	...	True	False	False	False	False	False	False	False	False	True
3	61	1	148	203	161	0.0	0	False	False	False	...	False	False	True	True	False	False	False	False	False	True
4	62	0	138	294	106	1.9	0	False	False	False	...	False	True	False	False	False	True	False	False	True	False

5 rows × 23 columns

```
In [18]: from sklearn.preprocessing import StandardScaler
         std = StandardScaler()
         df[num_val] = std.fit_transform(df[num_val])

In [19]: df.head(5)
```

Out[19]:

	age	sex	trestbps	chol	thalach	oldpeak	target	cp_1	cp_2	cp_3	...	exang_1	slope_1	slope_2	ca_1	ca_2	ca_3	ca_4	thal_1	thal_2	thal_3
0	-0.267966	1	-0.376556	-0.667728	0.806035	-0.037124	0	False	False	False	...	False	False	True	False	True	False	False	False	False	True
1	-0.157260	1	0.478910	-0.841918	0.237495	1.773958	0	False	False	False	...	True	False	False	False	False	False	False	False	False	True
2	1.724733	1	0.764066	-1.403197	-1.074521	1.342748	0	False	False	False	...	True	False	False	False	False	False	False	False	False	True
3	0.728383	1	0.935159	-0.841918	0.499898	-0.899544	0	False	False	False	...	False	False	True	True	False	False	False	False	False	True
4	0.839089	0	0.364848	0.919336	-1.905464	0.739054	0	False	False	False	...	False	True	False	False	False	True	False	False	True	False

5 rows × 23 columns

```
In [6]: #Spliting the dataset into the trainig set and test set
        X = df.drop('target', axis=1)
        Y= df['target']

In [7]: from sklearn.model_selection import train_test_split
        X_train , X_test , Y_train , Y_test  = train_test_split(X,Y ,test_size=0.2,random_state=42)

In [6]: Y_test

Out[6]: 527    1
        359    1
        447    0
        31     1
        621    0
              ..
        832    1
        796    1
        644    1
        404    0
        842    0
        Name: target, Length: 205, dtype: int64

In [7]: #logical Rergression
        df.head()
```

Out[7]:

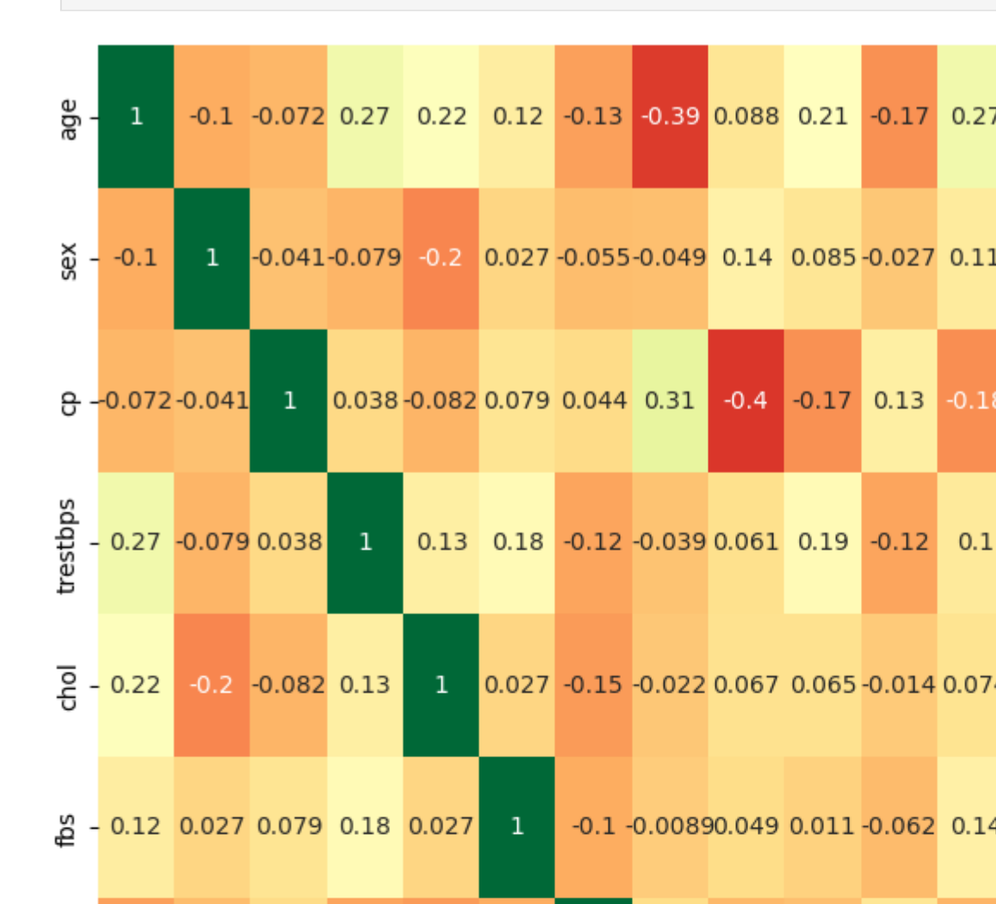
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
In [8]: from sklearn.linear_model import LogisticRegression
        log = LogisticRegression()
        log.fit(X_train , Y_train)
        y_predict1 = log.predict(X_test)
        from sklearn.metrics import accuracy_score
        accuracy_score(Y_test , y_predict1)

C:\Users\bshak\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=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[8]: 0.7853658536585366

In [9]: #Support Vector Classifier
        from sklearn import svm
        svm = svm.SVC()
        svm.fit(X_train , Y_train)
        y_predict2= svm.predict(X_test)
        accuracy_score(Y_test, y_predict2)

Out[9]: 0.6829268292682927

In [10]: #KNN neighbors
         from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier()
         knn.fit(X_train , Y_train)
         ypredict3 = knn.predict(X_test)
         accuracy_score(Y_test , ypredict3)

Out[10]: 0.7317073170731707

In [37]: pip install --upgrade scikit-learn

Requirement already satisfied: scikit-learn in c:\users\bshak\appdata\local\programs\python\python310\lib\site-packages (1.3.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\bshak\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn) (1.25.1)
Requirement already satisfied: scipy>=1.5.0 in c:\users\bshak\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\bshak\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn) (1.3.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\bshak\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn) (3.2.0)
Note: you may need to restart the kernel to use updated packages.

In [38]: pip install --upgrade numpy cython

Requirement already satisfied: numpy in c:\users\bshak\appdata\local\programs\python\python310\lib\site-packages (1.25.1)
Collecting cython
  Obtaining dependency information for cython from https://files.pythonhosted.org/packages/6d/0b/889b9b839ea7237eb6048191fe653c17ce93e298495eaf8f893cff748951/Cython-3.0.0-cp310-cp310-win_amd64.whl.metadata
  Using cached Cython-3.0.0-cp310-cp310-win_amd64.whl.metadata (3.2 kB)
Using cached Cython-3.0.0-cp310-cp310-win_amd64.whl (2.8 MB)
Installing collected packages: cython
Successfully installed cython-3.0.0
Note: you may need to restart the kernel to use updated packages.

In [11]: score = []
         for k in range (1,40):
             knn = KNeighborsClassifier(n_neighbors =k)
             knn.fit(X_train, Y_train)
             y_predict3= knn.predict(X_test)
             score.append(accuracy_score(Y_test, y_predict3))

In [12]: score

Out[12]: [0.9853658536585366,
          0.9560975609756097,
          0.9024390243902439,
          0.8097560975609757,
          0.7317073170731707,
          0.6878048780487804,
          0.6780487804878049,
          0.6682926829268293,
          0.7170731707317073,
          0.7170731707317073,
          0.6878048780487804,
          0.6780487804878049,
          0.7170731707317073,
          0.7073170731707317,
          0.6926829268292682,
          0.6780487804878049,
          0.7073170731707317,
          0.7121951219512195,
          0.7170731707317073,
          0.7073170731707317,
          0.6926829268292682,
          0.6829268292682927,
          0.6780487804878049,
          0.6926829268292682,
          0.7024390243902439,
          0.7024390243902439,
          0.6926829268292682,
          0.697560975609756,
          0.6731707317073171,
          0.6682926829268293,
          0.6439024390243903,
          0.6487804878048781,
          0.6390243902439025,
          0.6585365853658537,
          0.6536585365853659,
          0.6585365853658537,
          0.6341463414634146,
          0.6634146341463415]

In [14]: knn = KNeighborsClassifier(n_neighbors =2)
         knn.fit(X_train, Y_train)
         y_predict3 = knn.predict(X_test)
         accuracy_score(Y_test  , y_predict3)

Out[14]: 0.9560975609756097

In [15]: #RANDOM FOREST
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score
         rft = RandomForestClassifier()
         rft.fit(X_train, Y_train)
         y_predict4= rft.predict(X_test)
         accuracy_score(Y_test, y_predict4)

Out[15]: 0.9853658536585366

In [16]: #gradient boosting
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.metrics import accuracy_score
         grd = GradientBoostingClassifier()
         grd.fit(X_train, Y_train)
         y_predict5= grd.predict(X_test)
         accuracy_score(Y_test, y_predict5)

Out[16]: 0.9317073170731708

In [17]: new_data = pd.DataFrame({'Models':['LB','SVM','KNN','RF','GB'],
                                  'Accuracy':[accuracy_score(Y_test, y_predict1),
                                              accuracy_score(Y_test, y_predict2),
                                              accuracy_score(Y_test, y_predict3),
                                              accuracy_score(Y_test, y_predict4),
                                              accuracy_score(Y_test, y_predict5)
                                              ]})

In [18]: new_data
```

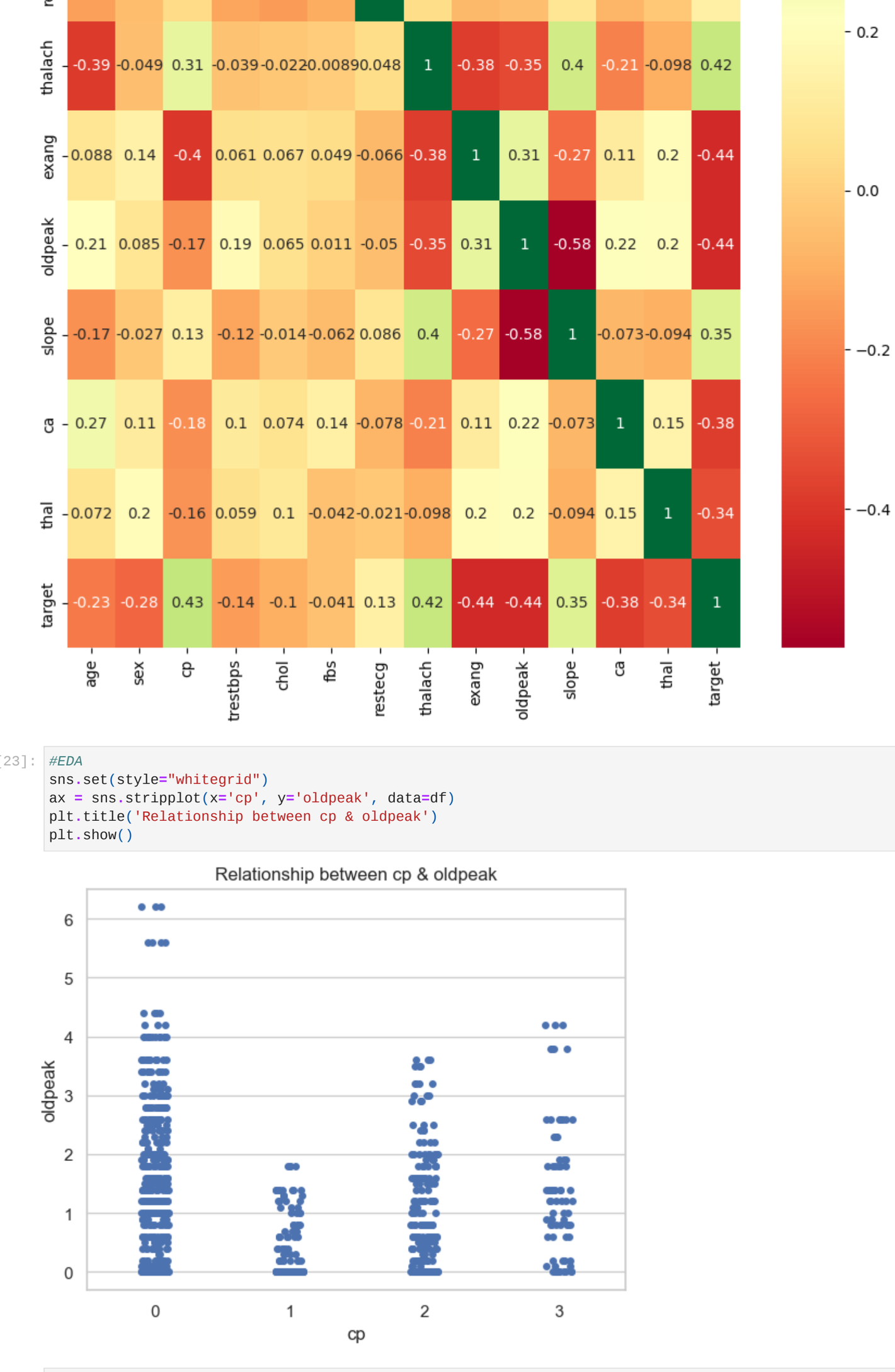
Out[18]:

	Models	Accuracy
0	LB	0.785366
1	SVM	0.682927
2	KNN	0.956098
3	RF	0.985366
4	GB	0.931707

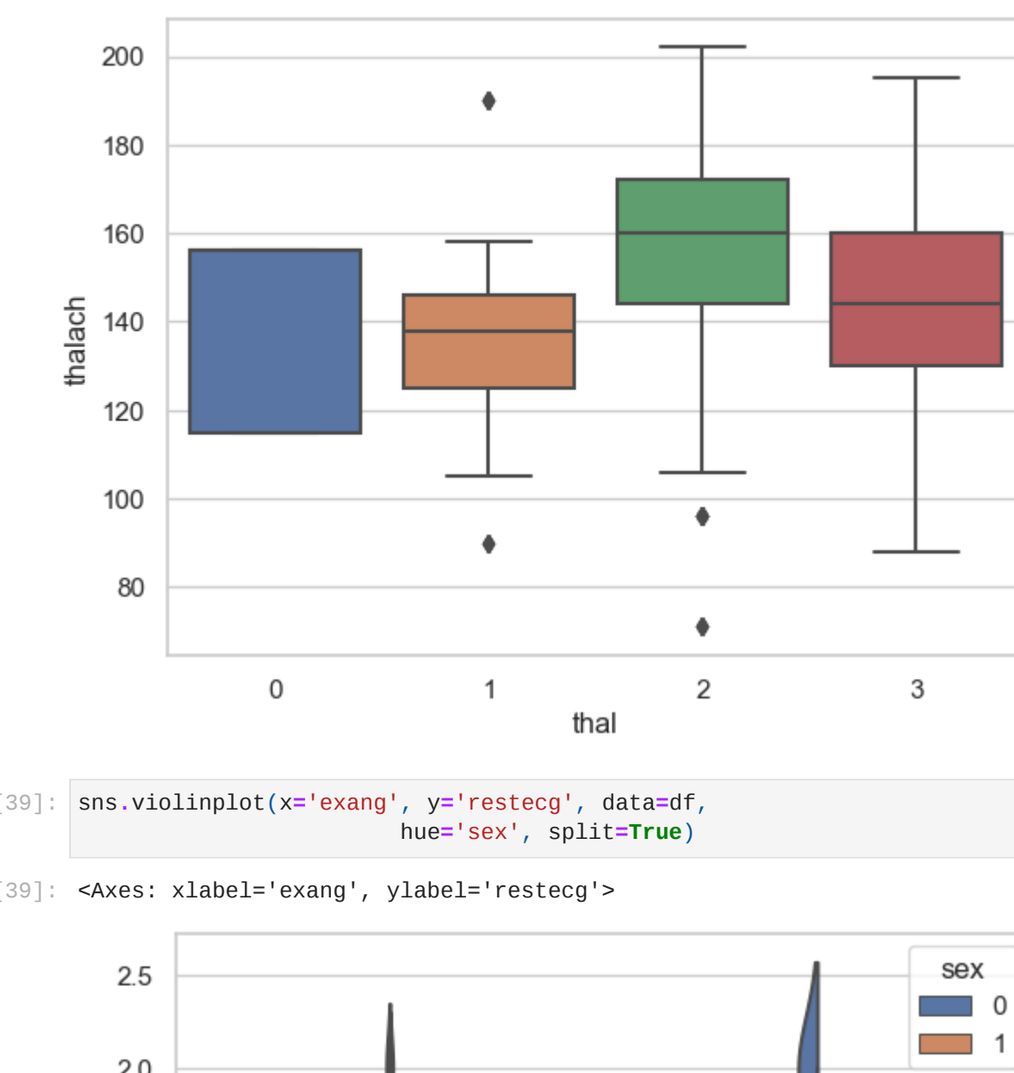
```
In [19]: sns.barplot(x=new_data['Models'] ,y = new_data['Accuracy'])

Out[19]: <Axes: xlabel='Models', ylabel='Accuracy'>
```



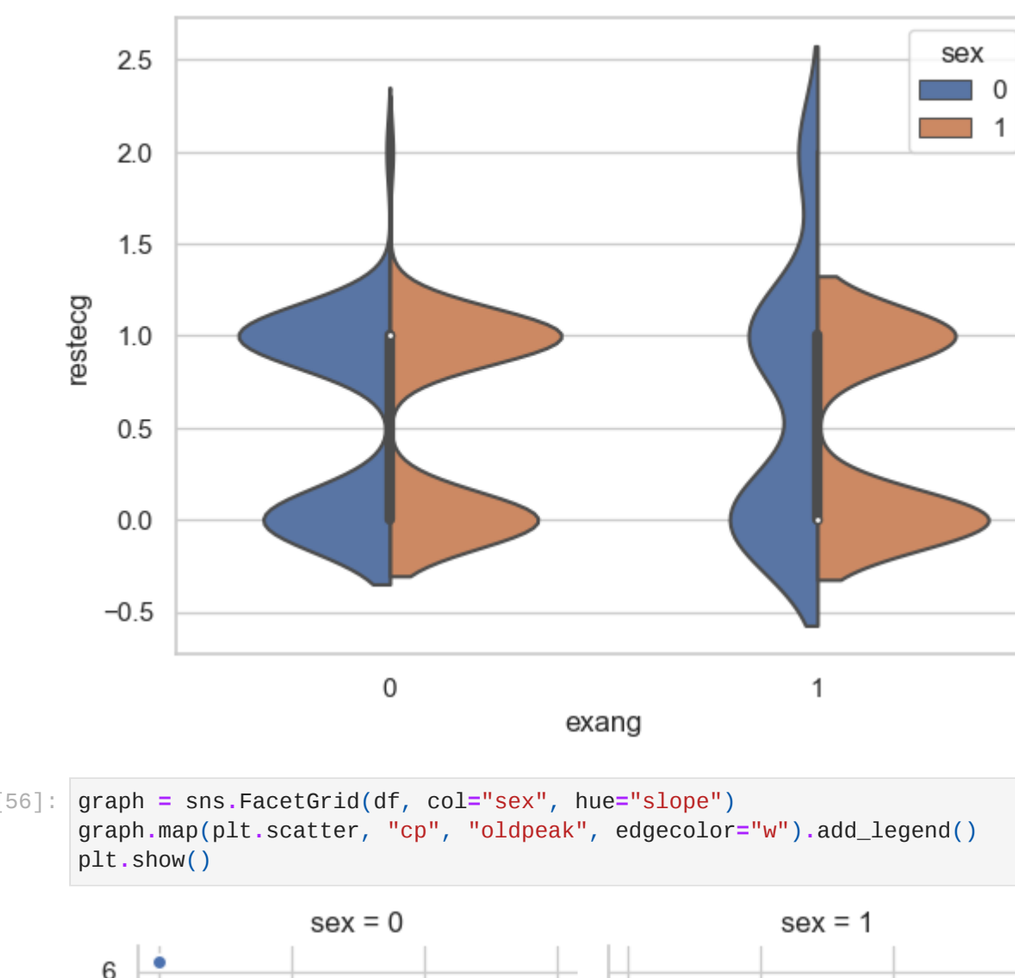

```
In [22]: #correlation
         corr = df.corr()
         top_corr_feature= corr.index
         plt.figure(figsize=(10,15))
         g = sns.heatmap(df[top_corr_feature].corr(), annot=True,cmap="RdYlGn")
         plt.show()
```



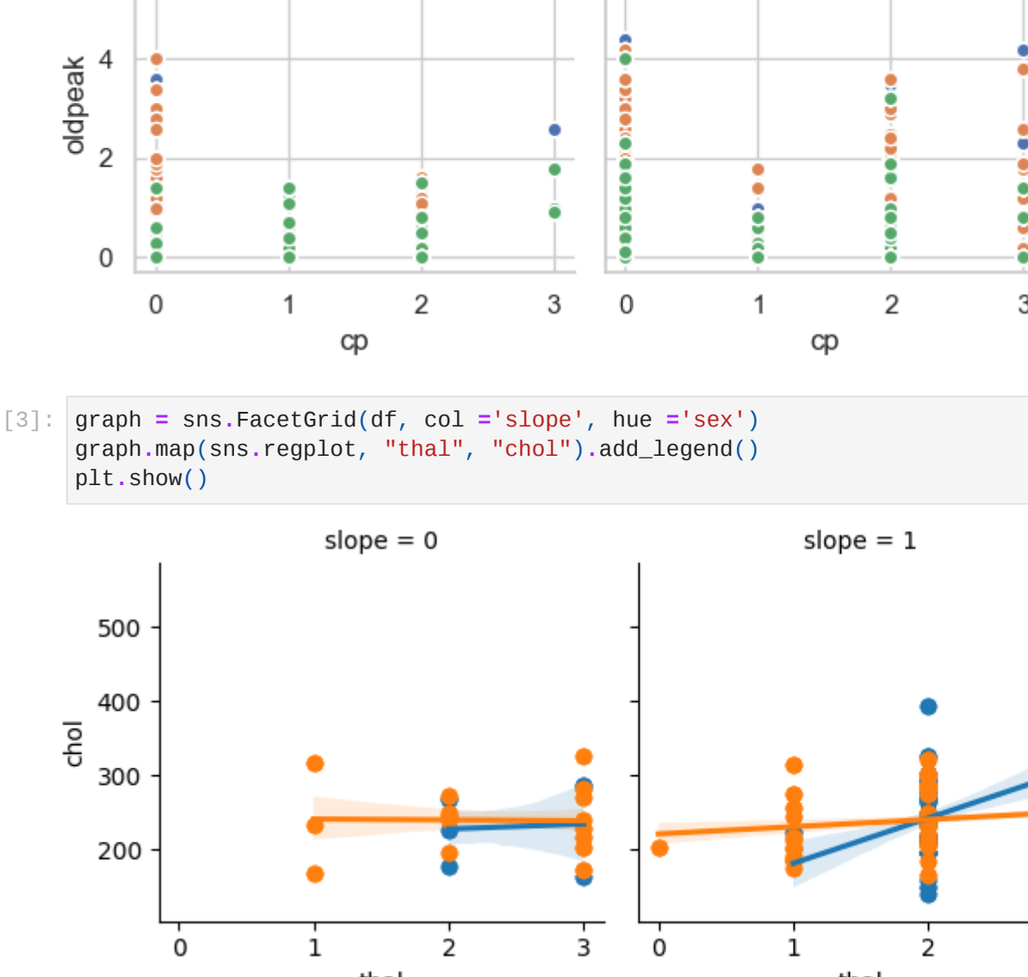

```
In [23]: #EDA
         sns.set(style="whitegrid")
         ax = sns.stripplot(x='cp', y='oldpeak', data=df)
         plt.title('Relationship between cp & oldpeak')
         plt.show()
```



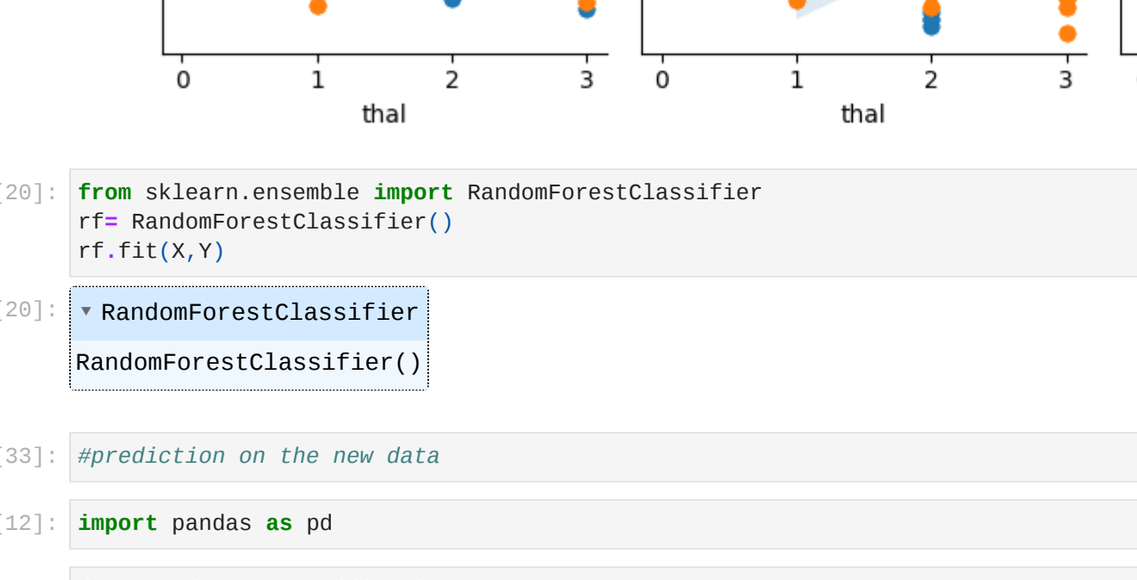

```
In [33]: sns.boxplot(x='thal', y='thalach',data=df)
         plt.show()
```



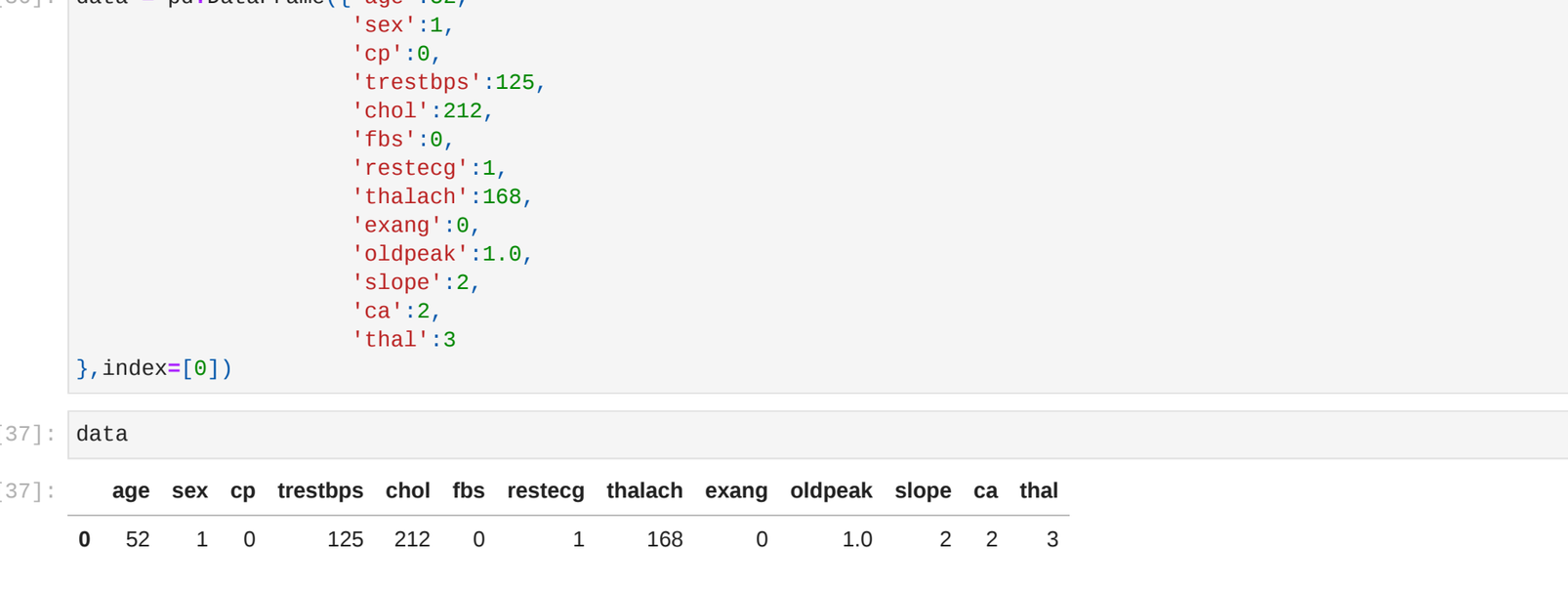

```
In [39]: sns.violinplot(x='exang', y='restecg', data=df,
                        hue='sex', split=True)

Out[39]: <Axes: xlabel='exang', ylabel='restecg'>
```




```
In [56]: graph = sns.FacetGrid(df, col="sex", hue="slope")
         graph.map(plt.scatter, "cp", "oldpeak", edgecolor="w").add_legend()
         plt.show()
```




```
In [3]: graph = sns.FacetGrid(df, col ='slope', hue ='sex')
        graph.map(sns.regplot, "thal", "chol").add_legend()
        plt.show()
```




```
In [20]: from sklearn.ensemble import RandomForestClassifier
         rf= RandomForestClassifier()
         rf.fit(X,Y)

Out[20]: ▾ RandomForestClassifier
         RandomForestClassifier()

In [33]: #prediction on the new data

In [12]: import pandas as pd

In [36]: data = pd.DataFrame({'age':52,
                              'sex':1,
                              'cp':0,
                              'trestbps':125,
                              'chol':212,
                              'fbs':0,
                              'restecg':1,
                              'thalach':168,
                              'exang':0,
                              'oldpeak':1.0,
                              'slope':2,
                              'ca':2,
                              'thal':3
         },index=[0])

In [37]: data
```

Out[37]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3

```
In [40]: p = rf.predict(data)
         if p[0]==0:
             print("No Disease")
         else:
             print("Disease")

No Disease

In [41]: #saving model using joblib

In [42]: import joblib

In [43]: joblib.dump(rf,'model_heart')

Out[43]: ['model_heart']

In [44]:  m = joblib.load('model_heart')

In [46]: m.predict(data)

Out[46]: array([0], dtype=int64)

In [21]: #Graphival user interface

In [22]: from tkinter import *
         import joblib

In [23]: import joblib
         import numpy as np
         from tkinter import *

         def entry_fields():
             # Retrieve the values from the entry widgets and convert them to integers
             p1 = int(e1.get())
             p2 = int(e2.get())
             p3 = int(e3.get())
             p4 = int(e4.get())
             p5 = int(e5.get())
             p6 = int(e6.get())
             p7 = int(e7.get())
             p8 = int(e8.get())
             p9 = int(e9.get())
             p10 = int(e10.get())
             p11 = int(e11.get())
             p12 = int(e12.get())
             p13 = int(e13.get())

             # Convert the input data to a 2D array with a single row
             input_data = np.array([[p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13]])

             # Load the machine learning model
             m = joblib.load('model_heart')

             # Make a prediction using the loaded model and the input features
             r = m.predict(input_data)

             # Display the prediction result
             if r == 0:
                 Label(master, text="No Heart disease").grid(row=31)
             else:
                 Label(master, text="Possibility of Heart disease").grid(row=31)

         # Create the main application window
         master = Tk()
         master.title("Heart disease prediction system")
         label = Label(master , text = "heart disease prediction system",
                       bg= "red", fg="white").grid(row=0 , columnspan =2)
         Label(master, text ="Enter your age").grid(row=1)
         Label(master, text ="Male or Female").grid(row=2)
         Label(master, text ="Enter CP value").grid(row=3)
         Label(master, text ="Enter value of trestbps").grid(row=4)
         Label(master, text ="Enter value of chol").grid(row=5)
         Label(master, text ="Enter value of fbs").grid(row=6)
         Label(master, text ="Enter value of restecg ").grid(row=7)
         Label(master, text ="Enter value of thalach ").grid(row=8)
         Label(master, text ="Enter value of exang").grid(row=9)
         Label(master, text ="Enter value of oldpeak").grid(row=10)
         Label(master, text ="Enter value of slope ").grid(row=11)
         Label(master, text ="Enter value of ca").grid(row=12)
         Label(master, text ="Enter value of thal").grid(row=13)

         e1 = Entry(master)
         e2 = Entry(master)
         e3 = Entry(master)
         e4 = Entry(master)
         e5 = Entry(master)
         e6 = Entry(master)  # Mistake in the original code, should be e6 instead of e7
         e7 = Entry(master)
         e8 = Entry(master)
         e9 = Entry(master)
         e10 = Entry(master)
         e11 = Entry(master)
         e12 = Entry(master)
         e13= Entry(master)

         e1.grid(row=1, column=1)
         e2.grid(row=2, column=1)
         e3.grid(row=3, column=1)
         e4.grid(row=4, column=1)
         e5.grid(row=5, column=1)
         e6.grid(row=6, column=1)
         e7.grid(row=7, column=1)
         e8.grid(row=8, column=1)
         e9.grid(row=9, column=1)
         e10.grid(row=10, column=1)
         e11.grid(row=11, column=1)
         e12.grid(row=12, column=1)
         e13.grid(row=13, column=1)

         Button(master, text='Predict', command =entry_fields).grid()
         mainloop()

C:\Users\bshak\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```