In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [2]:

```
df = pd.read_csv('adult.csv')
df.head()
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fe

In [3]:

```
df.columns
```

Out[3]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'country', 'salar
y'],
      dtype='object')
```

In [4]:

```
df.shape
```

Out[4]:

```
(32561, 15)
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
country           0
salary            0
dtype: int64
```

From the above data we can get that no null values in the dataset. But we have seen some duplicate values in the dataset

In [6]:

```
duplicates = df.duplicated().sum()
```

In [7]:

```
df = df.drop(duplicates)
```

In [8]:

```
df.shape
```

Out[8]:

```
(32560, 15)
```

In [9]:

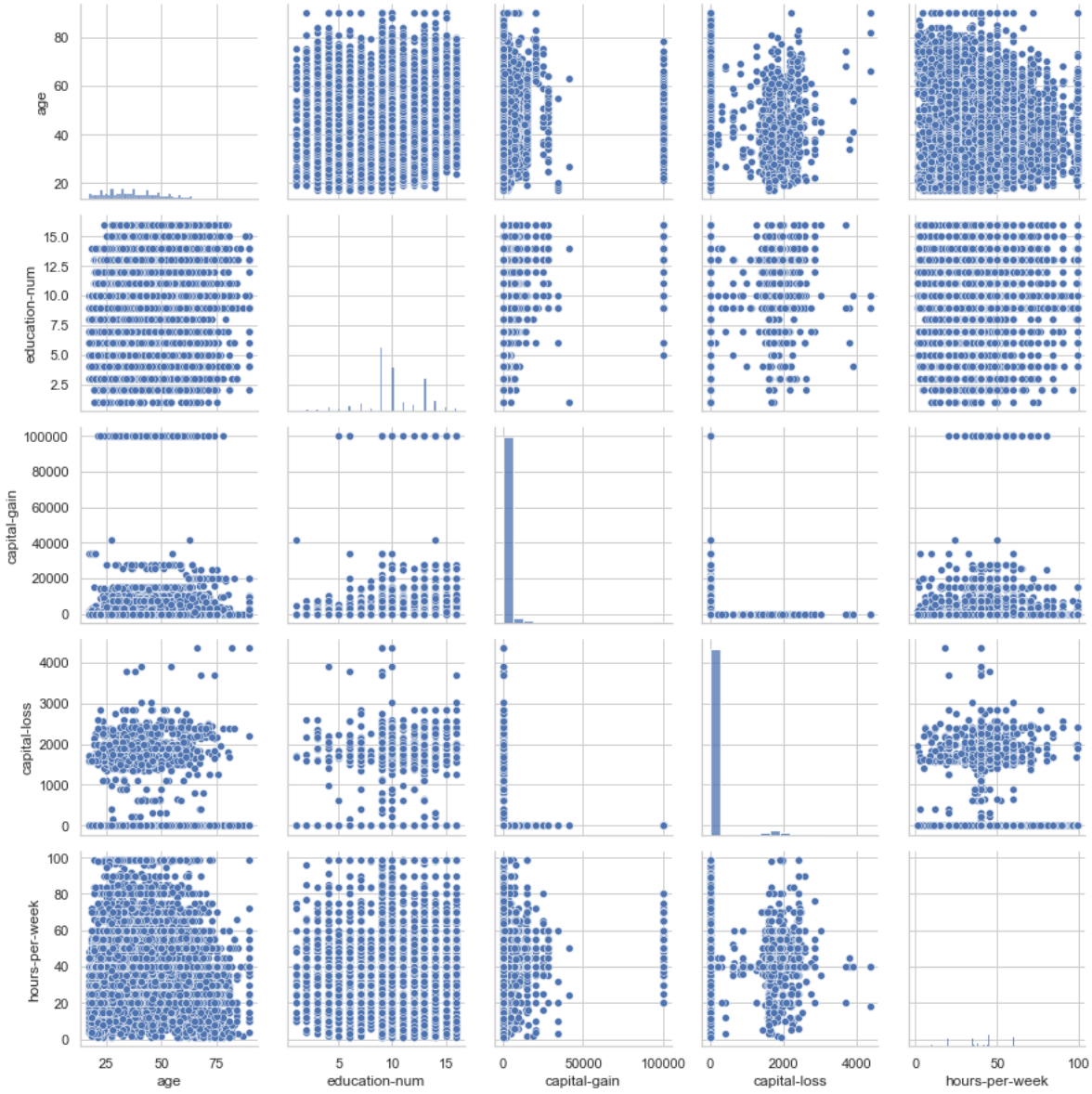
```
df.describe()
```

Out[9]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32560.000000	3.256000e+04	32560.000000	32560.000000	32560.000000	32560.000000
mean	38.581020	1.897808e+05	10.080713	1077.681941	87.306511	40.437469
std	13.640173	1.055506e+05	2.572753	7385.403083	402.966116	12.347618
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

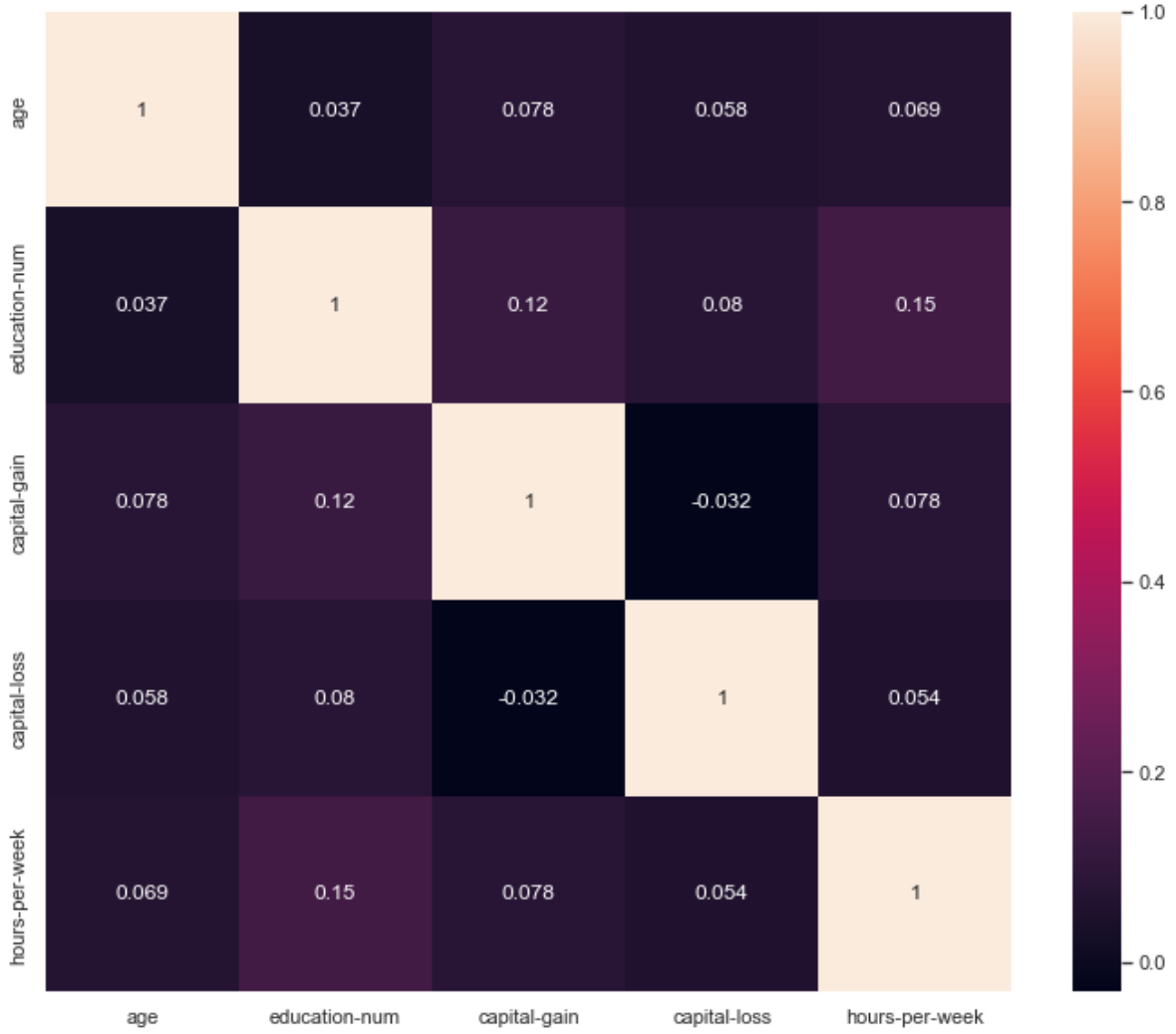
In [38]:

```
import seaborn as sns
sns.pairplot(df, height=2.5)
plt.tight_layout()
```



In [40]:

```
plt.figure(figsize=(12,10))
p=sns.heatmap(df.corr(), annot=True)
```

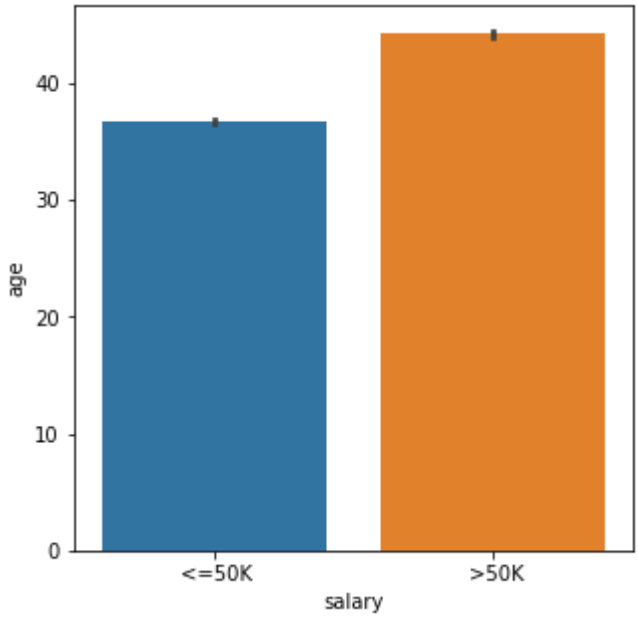


In [10]:

```
import matplotlib
matplotlib.rcParams['figure.figsize'] = (5, 5)
sns.barplot(y="age",x="salary",data=df)
```

Out[10]:

```
<AxesSubplot:xlabel='salary', ylabel='age'>
```

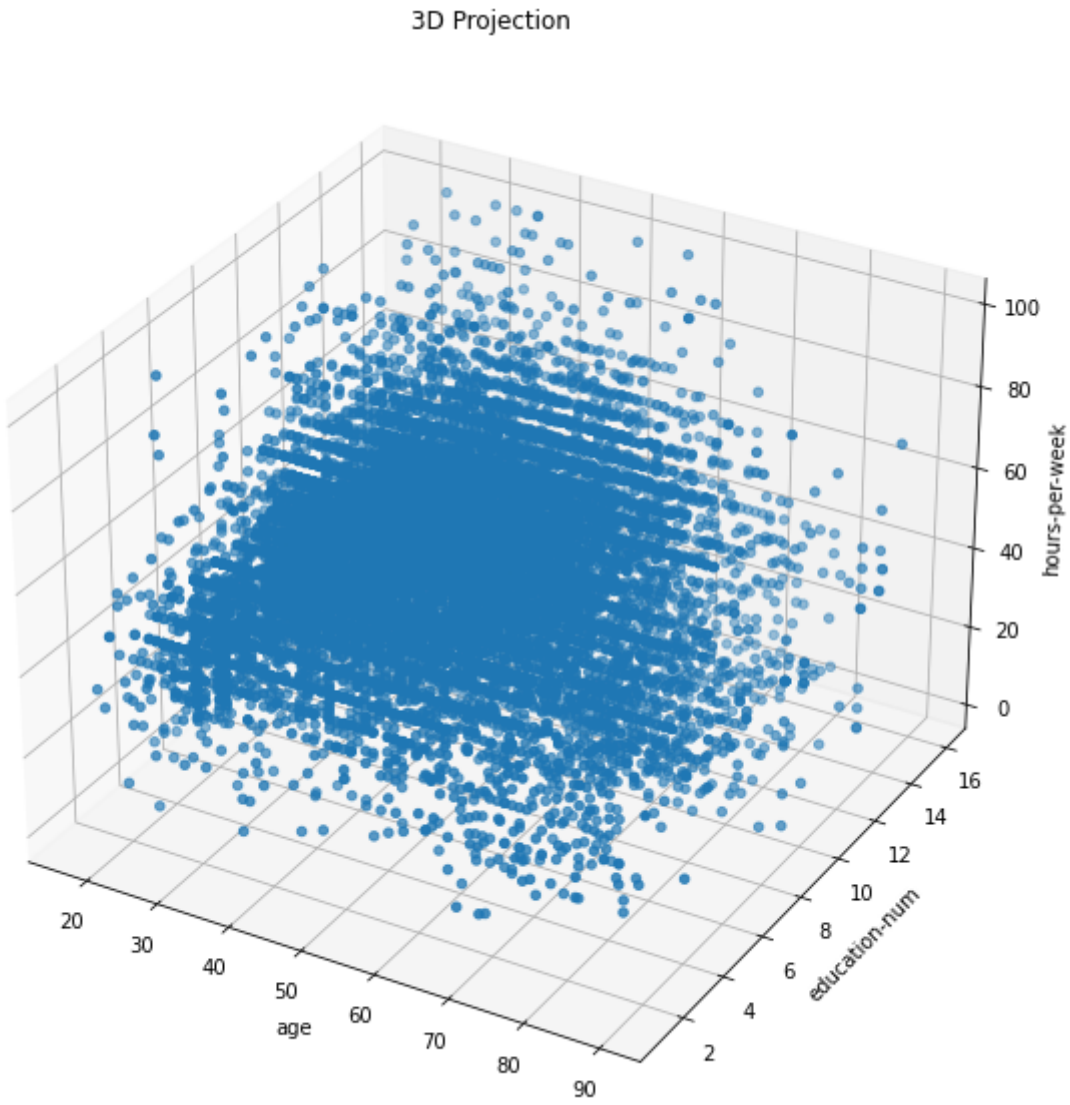


In [11]:

```
from mpl_toolkits.mplot3d import Axes3D
f=plt.figure(figsize=(10,10))
X=f.add_subplot(111,projection='3d')
X.scatter(df['age'],df['education-num'],df['hours-per-week'])
X.set_xlabel('age')
X.set_ylabel('education-num')
X.set_zlabel('hours-per-week')
X.set_title('3D Projection')
```

Out[11]:

```
Text(0.5, 0.92, '3D Projection')
```

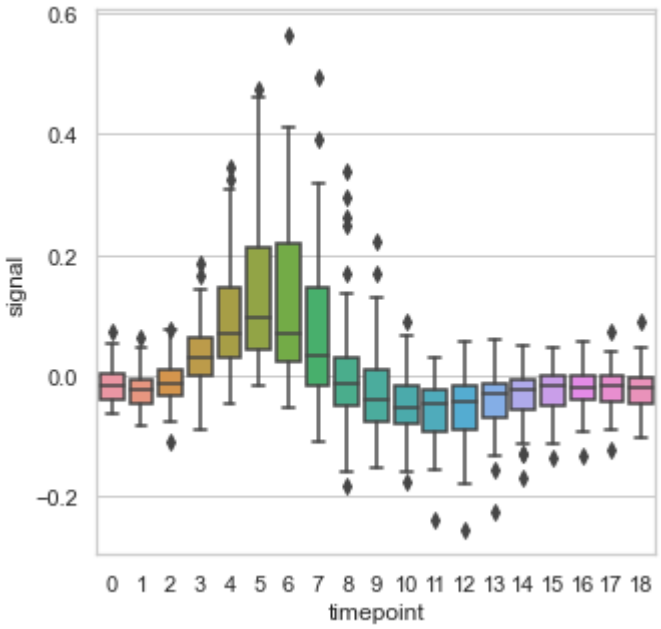


In [12]:

```
sns.set(style='whitegrid')
fmri=pd.read_csv("adult.csv")
fmri = sns.load_dataset("fmri")
sns.boxplot(x="timepoint",y="signal",data=fmri)
```

Out[12]:

```
<AxesSubplot:xlabel='timepoint', ylabel='signal'>
```



In [13]:

```
X = df.drop(['salary'], axis=True)
```

In [14]:

```
X.head()
```

Out[14]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fe

In [15]:

```
y = df['salary']
```

In [16]:

```
y.head()
```

Out[16]:

```
0     <=50K
1     <=50K
2     <=50K
3     <=50K
4     <=50K
Name: salary, dtype: object
```

In [17]:

```
y = pd.get_dummies(y)
```

In [18]:

```
y
```

Out[18]:

	<=50K	>50K
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...	...	...
32556	1	0
32557	0	1
32558	1	0
32559	1	0
32560	0	1

32560 rows × 2 columns

In [19]:

```
columns_to_use = ['age','workclass','education-num','sex','capital-gain','capital-loss','ho
```

In [20]:

```
df.columns
```

Out[20]:

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'country', 'salar
y'],
      dtype='object')
```

In [21]:

```
df = df[columns_to_use]
```

In [22]:

```
df.head()
```

Out[22]:

	age	workclass	education-num	sex	capital-gain	capital-loss	hours-per-week	country	salary
0	39	State-gov	13	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	13	Male	0	0	13	United-States	<=50K
2	38	Private	9	Male	0	0	40	United-States	<=50K
3	53	Private	7	Male	0	0	40	United-States	<=50K
4	28	Private	13	Female	0	0	40	Cuba	<=50K

In [23]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32560 entries, 0 to 32560
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32560 non-null  int64 
 1   workclass       32560 non-null  object
 2   education-num   32560 non-null  int64 
 3   sex             32560 non-null  object
 4   capital-gain    32560 non-null  int64 
 5   capital-loss    32560 non-null  int64 
 6   hours-per-week  32560 non-null  int64 
 7   country         32560 non-null  object
 8   salary          32560 non-null  object
dtypes: int64(5), object(4)
memory usage: 3.5+ MB
```

In [24]:

```
df.nunique()
```

Out[24]:

```
age               73
workclass          9
education-num     16
sex                2
capital-gain     119
capital-loss      92
hours-per-week    94
country           42
salary             2
dtype: int64
```

In [25]:

```
X = df.drop(['salary'],axis=1)
```

In [26]:

```
X.head()
```

Out[26]:

	age	workclass	education-num	sex	capital-gain	capital-loss	hours-per-week	country
0	39	State-gov	13	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	13	Male	0	0	13	United-States
2	38	Private	9	Male	0	0	40	United-States
3	53	Private	7	Male	0	0	40	United-States
4	28	Private	13	Female	0	0	40	Cuba

In [27]:

```
y = df['salary']
```

In [28]:

```
y.head
```

Out[28]:

```
<bound method NDFrame.head of 0         <=50K
1         <=50K
2         <=50K
3         <=50K
4         <=50K
          ...
32556     <=50K
32557      >50K
32558     <=50K
32559     <=50K
32560      >50K
Name: salary, Length: 32560, dtype: object>
```

In [29]:

```
y = pd.get_dummies(y,drop_first=True)[' >50K']
y
```

Out[29]:

```
0        0
1        0
2        0
3        0
4        0
        ..
32556    0
32557    1
32558    0
32559    0
32560    1
Name:  >50K, Length: 32560, dtype: uint8
```

In [30]:

```
X = pd.get_dummies(X)
```

In [31]:

```
X.head()
```

Out[31]:

	age	education-num	capital-gain	capital-loss	hours-per-week	workclass_?	workclass_Federal-gov	workclass_Local-gov	workclass Neve worke
0	39	13	2174	0	40	0	0	0	
1	50	13	0	0	13	0	0	0	
2	38	9	0	0	40	0	0	0	
3	53	7	0	0	40	0	0	0	
4	28	13	0	0	40	0	0	0	

5 rows × 58 columns

In [32]:

```
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=1)
```

In [33]:

```
from sklearn.linear_model import LogisticRegression
Logistic_reg = LogisticRegression()
Logistic_reg.fit(x_train,y_train)
Logistic_reg.score(x_test,y_test)
```

```
D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: Converg
enceWarning: 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 (https://scik
it-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-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Out[33]:

```
0.8135749385749386
```

In [34]:

```
from sklearn.tree import DecisionTreeClassifier
Decision_Treee = DecisionTreeClassifier()
Decision_Treee.fit(x_train,y_train)
Decision_Treee.score(x_test,y_test)
```

Out[34]:

```
0.812039312039312
```

In [35]:

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
Random_classifier = RandomForestClassifier()
Random_classifier.fit(x_train,y_train)
Random_classifier.score(x_test,y_test)
```

Out[35]:

```
0.8249385749385749
```