Random Forest

In [1]:
```
import pandas as pd
from sklearn.datasets import load_digits
```

In [2]:
```
df = load_digits()
df
```

Out[2]:
```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
  'pixel_1_1',
```

In [3]:
```
import sklearn
df =sklearn.datasets.load_digits()
```

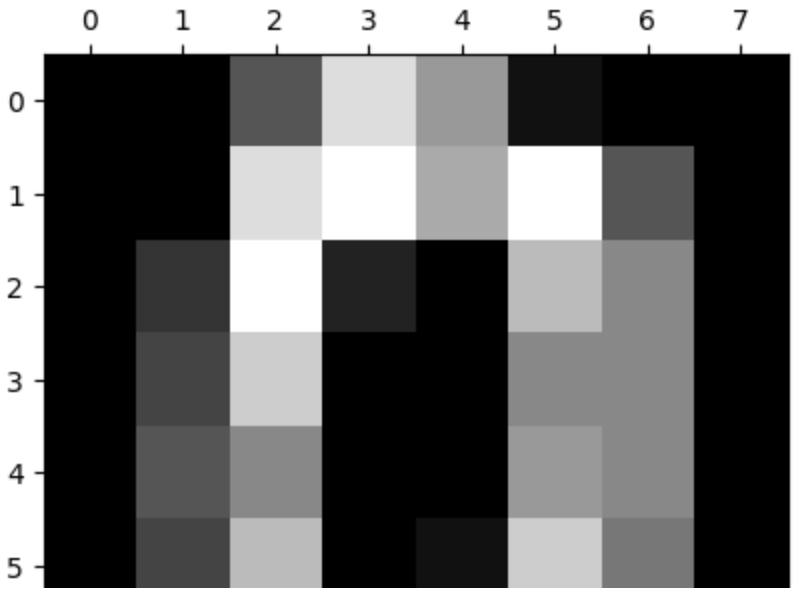
In [4]:
```
dir(df)
```

Out[4]:
```
['DESCR', 'data', 'feature_names', 'frame', 'images', 'target', 'target_name
s']
```

In [5]:
```
import matplotlib.pyplot as plt
```

In [6]:
```
# plt.figure(figsize=(4,3))
plt.gray()
for i in range(10):
    plt.matshow(df.images[i])
```

```
<Figure size 640x480 with 0 Axes>
```



In [7]:
```
df1= pd.DataFrame(df.data)
df1
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60	
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0	10.0	(
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0	16.0	1(
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0	11.0	1(
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	7.0	13.0	13.0	9
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	2.0	16.0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1792	0.0	0.0	4.0	10.0	13.0	6.0	0.0	0.0	0.0	1.0	...	4.0	0.0	0.0	0.0	2.0	14.0	15.0	9
1793	0.0	0.0	6.0	16.0	13.0	11.0	1.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	6.0	16.0	14.0	6
1794	0.0	0.0	1.0	11.0	15.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	9.0	13.0	6
1795	0.0	0.0	2.0	10.0	7.0	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0	5.0	12.0	16.0	1
1796	0.0	0.0	10.0	14.0	8.0	1.0	0.0	0.0	0.0	2.0	...	8.0	0.0	0.0	1.0	8.0	12.0	14.0	1

1797 rows × 64 columns

In [8]:
```
df.data
```

Out[8]:
```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

In [9]:
```
df1.sample(5)
```

Out[9]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60
1494	0.0	0.0	2.0	11.0	10.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0	3.0	13.0	14.0
12	0.0	0.0	5.0	12.0	1.0	0.0	0.0	0.0	0.0	0.0	...	8.0	2.0	0.0	0.0	3.0	11.0	8.0
504	0.0	0.0	2.0	8.0	8.0	8.0	12.0	2.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	15.0	0.0
269	0.0	0.0	6.0	15.0	16.0	10.0	0.0	0.0	0.0	3.0	...	5.0	0.0	0.0	0.0	9.0	16.0	16.0
1333	0.0	0.0	9.0	16.0	16.0	16.0	7.0	0.0	0.0	3.0	...	0.0	0.0	0.0	0.0	10.0	16.0	16.0

5 rows × 64 columns

In [10]:
```
df1.describe()
```

Out[10]:

	0	1	2	3	4	5	6	17
count	1797.0	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	
mean	0.0	0.303840	5.204786	11.835838	11.848080	5.781859	1.362270	
std	0.0	0.907192	4.754826	4.248842	4.287388	5.666418	3.325775	
min	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.0	0.000000	1.000000	10.000000	10.000000	0.000000	0.000000	
50%	0.0	0.000000	4.000000	13.000000	13.000000	4.000000	0.000000	
75%	0.0	0.000000	9.000000	15.000000	15.000000	11.000000	0.000000	
max	0.0	8.000000	16.000000	16.000000	16.000000	16.000000	16.000000	

8 rows × 64 columns

In [11]:
```
df1.sample(1)
```

Out[11]:

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60	61	62
350	0.0	0.0	6.0	15.0	16.0	3.0	0.0	0.0	0.0	3.0	...	0.0	0.0	0.0	0.0	7.0	9.0	0.0	0.0	0.0

1 rows × 64 columns

In [12]:
```
x = df1 ['target']=df.target
x
```

Out[12]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [13]:
```
x = df1.drop('target', axis='columns')
```

In [14]:
```
y =df1['target']
from sklearn.ensemble import RandomForestClassifier
```

In [15]:
```
model = RandomForestClassifier(n_estimators=25)
```

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train,  x_test, y_train, y_test= train_test_split(x,y)
```

In [18]:
```
x_train.shape
```

Out[18]:
```
(1347, 64)
```

In [19]:
```
x_test.shape
```

Out[19]:
```
(450, 64)
```

In [20]:
```
df1.shape
```

Out[20]:
```
(1797, 65)
```

In [21]:
```
model.fit(x_train, y_train)
```

Out[21]:
```
RandomForestClassifier
RandomForestClassifier(n_estimators=25)
```

In [22]:
```
pred= model.predict(x_test)
pred
```

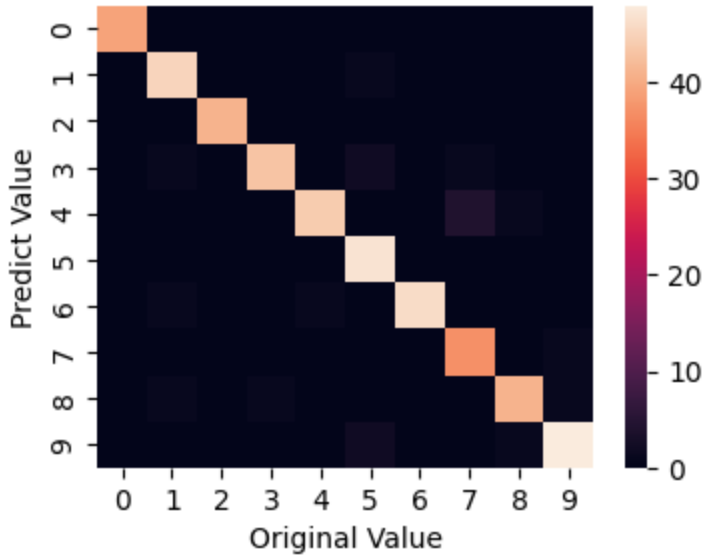
Out[22]:
```
array([9, 8, 8, 5, 5, 6, 0, 3, 3, 1, 3, 3, 4, 1, 0, 3, 2, 9, 7, 6, 4, 9,
       7, 8, 7, 0, 5, 7, 4, 5, 0, 4, 9, 2, 7, 1, 7, 0, 2, 8, 0, 9, 9, 8,
       4, 4, 0, 5, 1, 6, 7, 3, 9, 7, 4, 8, 1, 9, 1, 8, 1, 5, 1, 9, 4, 6,
       1, 2, 5, 8, 0, 3, 2, 0, 9, 2, 2, 2, 9, 3, 6, 4, 3, 8, 9, 3, 8, 2,
       5, 2, 4, 5, 0, 5, 7, 2, 4, 1, 8, 8, 1, 8, 9, 3, 1, 8, 5, 6, 1, 4,
       5, 5, 6, 3, 9, 3, 8, 3, 0, 3, 3, 6, 5, 2, 2, 9, 2, 7, 9, 4, 0, 1,
       5, 7, 1, 2, 1, 6, 9, 8, 4, 2, 5, 5, 1, 2, 8, 9, 0, 4, 6, 0, 2, 0,
       4, 1, 7, 3, 4, 1, 9, 6, 6, 9, 6, 2, 5, 1, 7, 7, 2, 3, 3, 3, 5, 4,
       8, 9, 0, 9, 1, 7, 8, 9, 0, 0, 9, 5, 6, 1, 3, 7, 0, 6, 6, 8, 1, 2,
       6, 5, 9, 6, 5, 4, 3, 3, 0, 1, 2, 7, 0, 6, 7, 6, 6, 2, 5, 8, 3, 1,
       4, 7, 0, 1, 2, 6, 0, 7, 6, 8, 1, 1, 3, 2, 6, 3, 9, 9, 5, 6, 7, 8,
       3, 3, 7, 7, 9, 0, 4, 3, 7, 8, 4, 8, 6, 7, 5, 4, 4, 4, 1, 1, 9, 5,
       7, 1, 5, 5, 2, 9, 4, 2, 3, 1, 2, 2, 9, 4, 5, 7, 5, 4, 1, 5, 5, 3,
       6, 3, 0, 4, 0, 0, 8, 9, 1, 4, 4, 7, 5, 2, 2, 5, 8, 8, 0, 6, 8, 6,
       4, 8, 5, 5, 6, 7, 1, 5, 7, 7, 3, 2, 8, 9, 6, 3, 9, 9, 6, 6, 3, 3,
       9, 5, 4, 6, 1, 7, 4, 0, 2, 0, 4, 9, 0, 6, 2, 9, 0, 9, 9, 3, 7, 9,
       7, 4, 8, 6, 6, 6, 5, 9, 4, 4, 8, 9, 3, 5, 4, 3, 9, 5, 2, 5, 2, 4,
       3, 6, 5, 7, 1, 9, 7, 4, 7, 9, 1, 5, 5, 6, 5, 6, 5, 6, 7, 4, 0, 1,
       7, 8, 1, 5, 8, 0, 0, 5, 4, 9, 2, 4, 1, 2, 9, 8, 5, 6, 8, 5, 3, 6,
       1, 7, 6, 6, 0, 9, 3, 0, 1, 0, 3, 0, 8, 4, 2, 8, 1, 7, 1, 1, 8, 8,
       2, 2, 1, 5, 3, 8, 9, 7, 8, 1])
```

In [23]:
```
y_pred = model.predict(x_test)
```

In [24]:
```
from sklearn.metrics import confusion_matrix
cm =confusion_matrix(y_test, y_pred)
cm
```

Out[24]:
```
array([[39,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 45,  0,  0,  0,  1,  0,  0,  0,  0],
       [ 0,  0, 41,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  0, 43,  0,  2,  0,  1,  0,  0],
       [ 0,  0,  0,  0, 44,  0,  0,  4,  1,  0],
       [ 0,  0,  0,  0,  0, 47,  0,  0,  0,  0],
       [ 0,  1,  0,  0,  1,  0, 46,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 37,  0,  1],
       [ 0,  1,  0,  1,  0,  0,  0,  0, 41,  1],
       [ 0,  0,  0,  0,  0,  2,  0,  0,  1, 48]], dtype=int64)
```

In [25]:
```
import seaborn as sns
plt.figure(figsize=(4,3))
sns.heatmap(cm)#annot=True, vmin=0, vmax=10
plt.xlabel('Original Value')
plt.ylabel('Predict Value')
plt.show()
```



In [26]:
```
df =pd.read_csv('die.csv')
df
```

Out[26]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns

In [27]:
```
df.sample(5)
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
447	0	95	80	45	92	36.5	0.33
244	2	146	76	35	194	38.2	0.32
764	2	122	70	27	0	36.8	0.34
140	3	128	78	0	0	21.1	0.26
428	0	135	94	46	145	40.6	0.28

In [28]:
```
df.info()
```

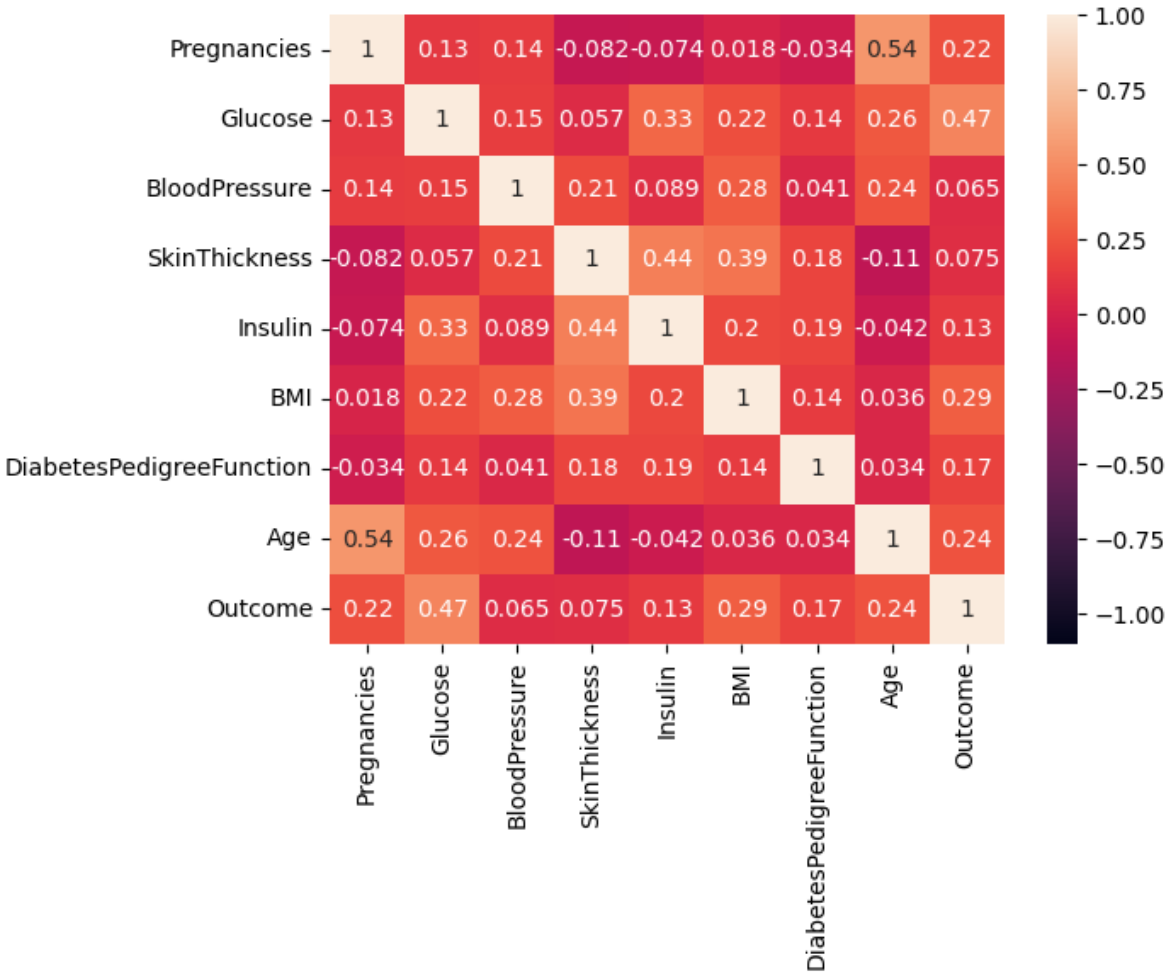
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [29]:
```
import seaborn as sns
sns.pairplot(df, hue='Outcome')
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x1236bd50090>

In [30]:
```
sns.heatmap(df.corr(), vmin=1, vmax=-1, annot=True)
```

Out[30]: <Axes: >



In [31]:
```
df.corr()
```

Out[31]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292

In [32]:
```
x =df.drop('Outcome', axis=1)
x.sample()
```

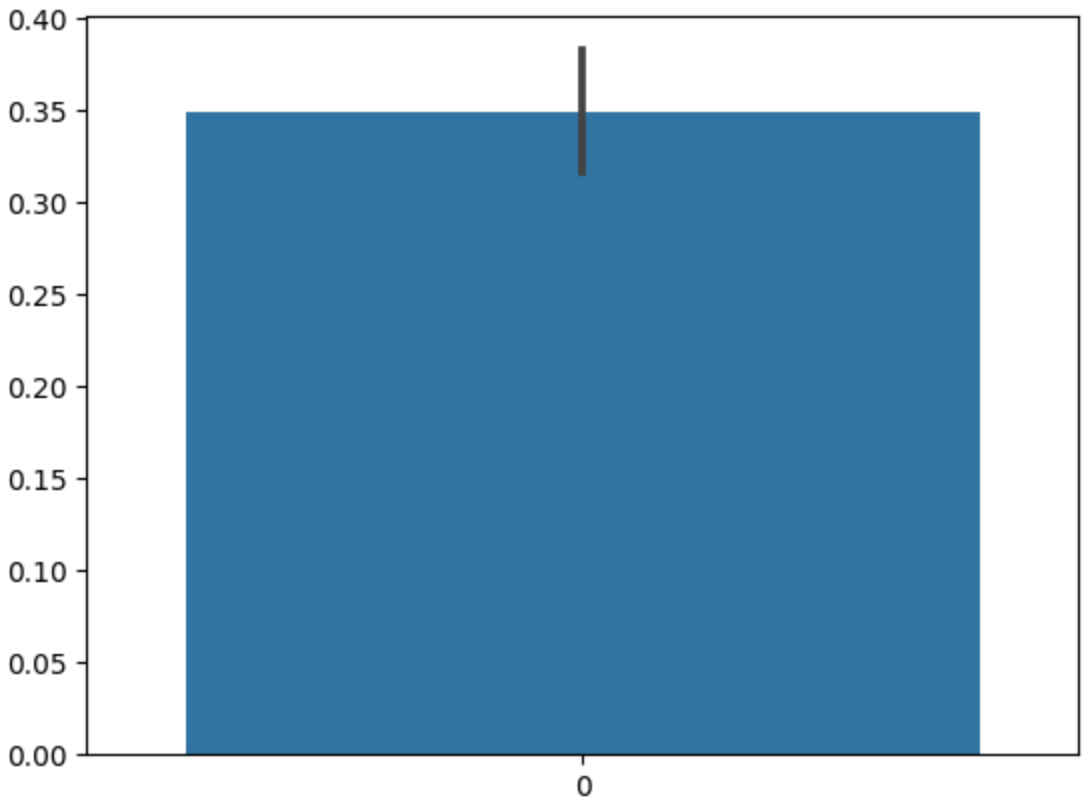
Out[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
343	5	122	86	0	0	34.7	0.2

In [33]:
```
y =df['Outcome']
```

In [34]:
```
import matplotlib.pyplot as plt
sns.barplot(y)
```

Out[34]: <Axes: >



In [35]:
```
from sklearn.model_selection import train_test_split
```

In [36]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y)
```

In [37]:
```
x_train.sample()
```

Out[37]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
571	2	130	96	0	0	22.6	0.26

In [38]:
```
y_train.sample()
```

Out[38]:
```
288    0
Name: Outcome, dtype: int64
```

In [39]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```

In [40]:
```
model.fit(x_train, y_train)
```

Out[40]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [41]:
```
y_pred = model.predict(x_test)
```

In [42]:
```
import numpy as np
a =np.array(y_test)
a
```

Out[42]:
```
array([1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0], dtype=int64)
```

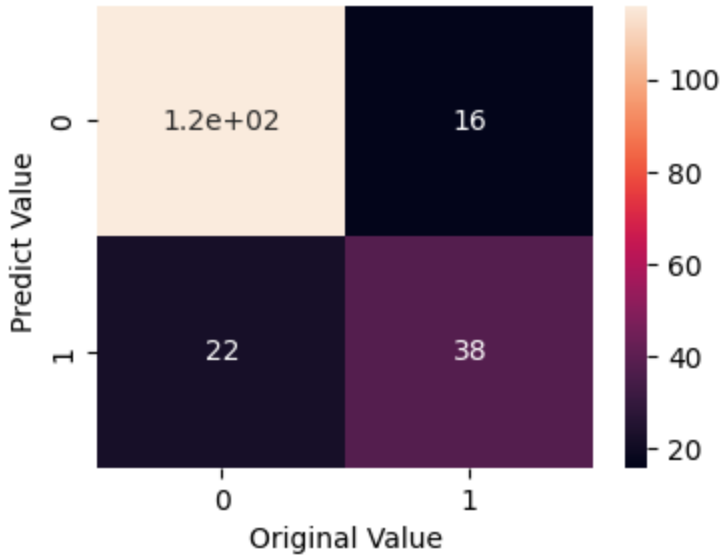
In [43]:
```
model.score(x_test,y_test)
```

Out[43]:
```
0.8020833333333334
```

In [44]:
```
from sklearn.metrics import confusion_matrix
cm =confusion_matrix(y_test, y_pred)
cm
```

Out[44]:
```
array([[116,  16],
       [ 22,  38]], dtype=int64)
```

In [45]:
```
plt.figure(figsize=(4,3))
sns.heatmap(cm, annot=True)
plt.xlabel('Original Value')
plt.ylabel('Predict Value ')
plt.show()
# countif(c1:c100,0)
```



In [46]:
```
x_test.shape
```

Out[46]:
```
(192, 8)
```

In [47]:
```
y_train.shape
```

Out[47]:
```
(576,)
```

In [48]:
```
x_train.shape
```

Out[48]:
```
(576, 8)
```

In [49]:
```
y_test.shape
```

Out[49]:
```
(192,)
```

In [50]:
```
from sklearn.metrics import classification_report
classification_report(y_test, y_pred)
```

Out[50]:
```
'              precision    recall  f1-score   support\n\n           0
0.84      0.88      0.86       132\n           1       0.70      0.63      0.
67        60\n\n    accuracy                           0.80       192\n   mac
ro avg       0.77      0.76      0.76       192\nweighted avg       0.80
0.80      0.80       192\n'
```

In [51]:
```
y_prob = model.predict_proba(x_test)[:,1]
# y_prob
```

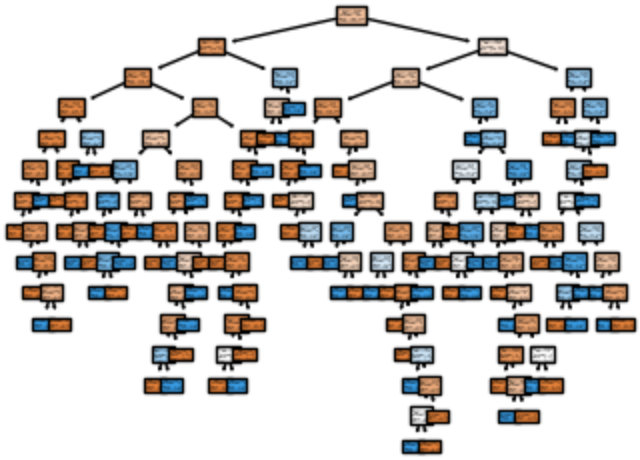
In [52]:
```
from sklearn.metrics import roc_auc_score
```

In [53]:
```
roc_auc_score(y_test, y_pred)
```

Out[53]: 0.7560606060606061

In [54]:
```
from sklearn.tree import plot_tree
```

In [55]:
```
plt.figure(figsize=(4,3))

# a =df['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI

plot_tree(model.estimators_[5], class_names=['Glucos','BMI'], filled=True)
plt.savefig('mm.png', dpi=750)
```



In [56]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [57]:
```
df =pd.read_csv('die.csv')
df
```

Out[57]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns

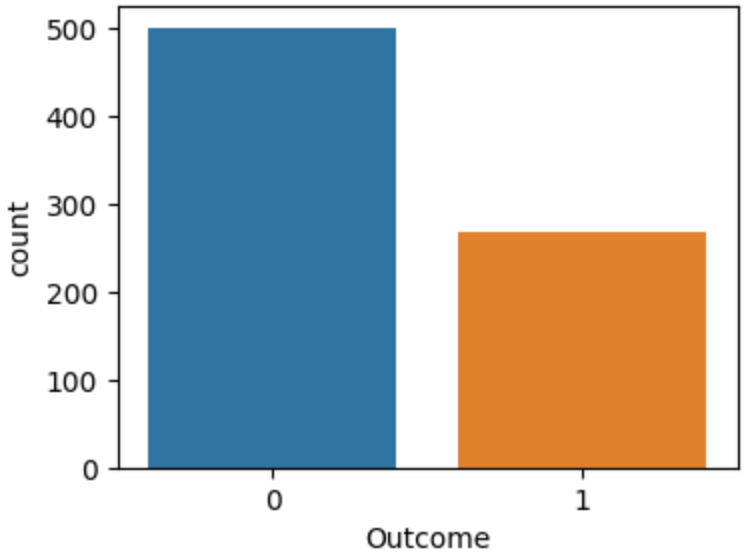
In [58]:
```
df.sample(4)
```

Out[58]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
341	1	95	74	21	73	25.9	0.67
200	0	113	80	16	0	31.0	0.87
662	8	167	106	46	231	37.6	0.16
112	1	89	76	34	37	31.2	0.19

In [59]:
```
# COUNTPLOT
plt.figure(figsize=(4,3))
sns.countplot(x='Outcome', data=df)
```

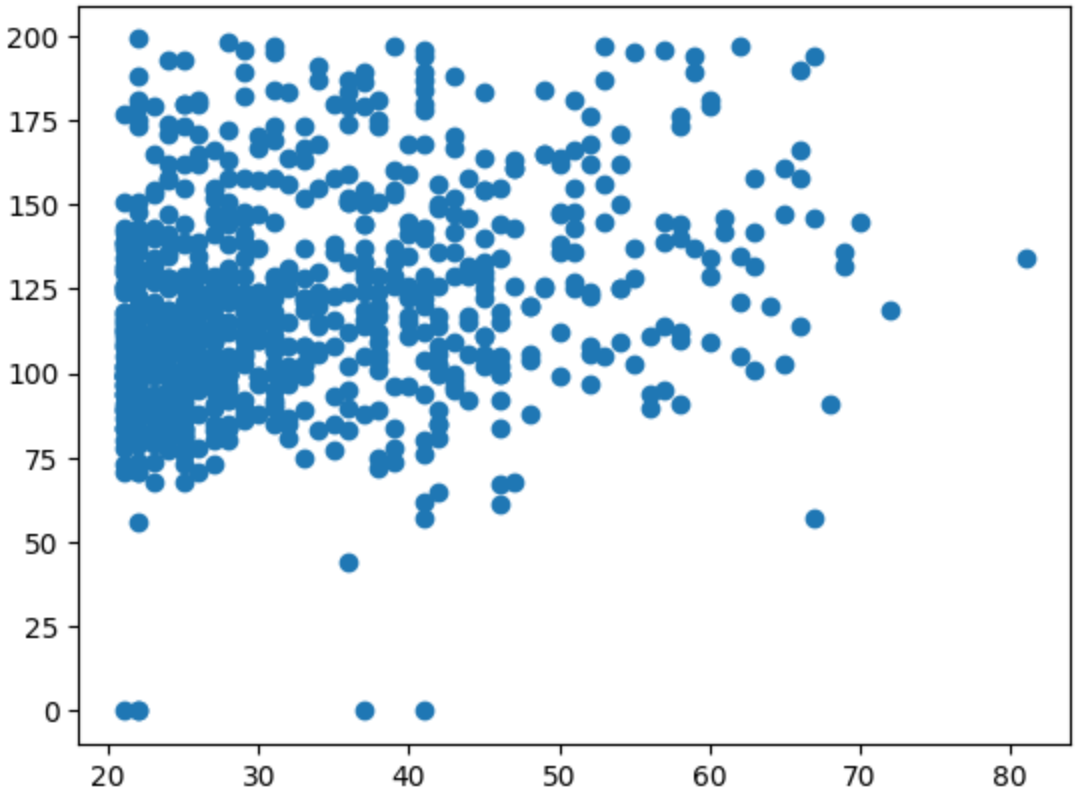
Out[59]: <Axes: xlabel='Outcome', ylabel='count'>



In [60]:
```
x =df['Age']
y = df['Glucose']
```

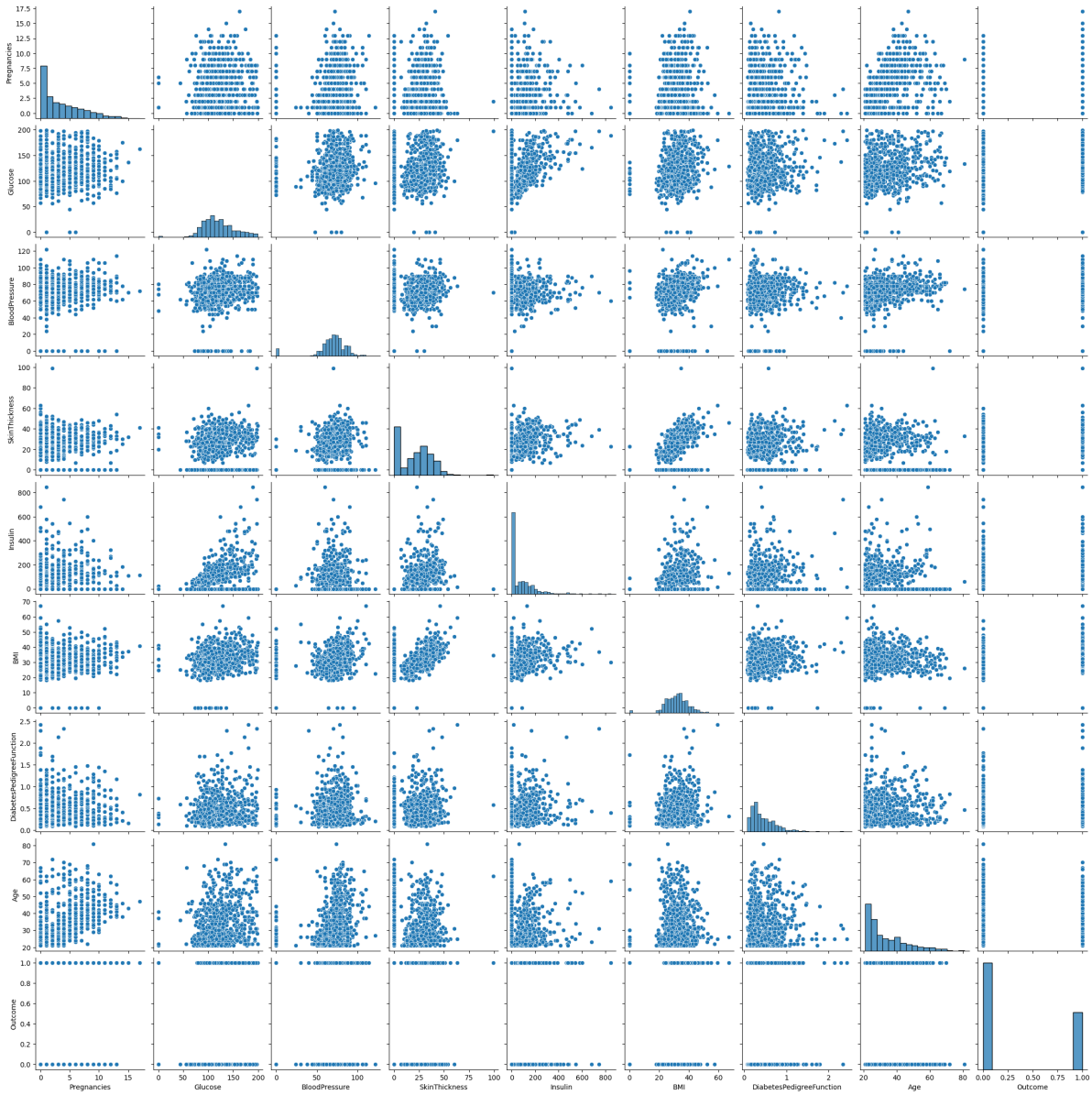
In [61]:
```
plt.scatter(x,y)
```

Out[61]:
```
<matplotlib.collections.PathCollection at 0x123734aaf10>
```



In [62]:
```
sns.pairplot(df)  # ---> multiple ploting
```

Out[62]:
```
<seaborn.axisgrid.PairGrid at 0x12372f580d0>
```



In [63]:
```
from sklearn.cluster import KMeans
```

In [64]:
```
#build Model
km = KMeans(n_clusters=2)
```

In [65]:
```
y_pred = km.fit_predict(df[['Age', 'BMI']])
y_pred
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_i
nit` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicit
ly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[65]:
```
array([1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0])
```

In [66]:
```
y.shape
# y_pred.shape
```

Out[66]:
```
(768,)
```

In [67]:
```
df['cluster']=y_pred
df
```

Out[67]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 10 columns

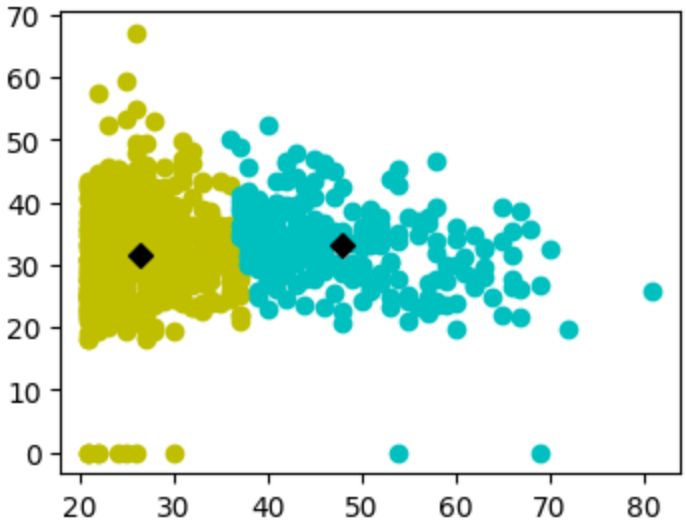
In [68]:
```
km.cluster_centers_
```

Out[68]:
```
array([[26.42748092, 31.43492366],
       [47.87295082, 33.19016393]])
```

In [69]:
```
df1 = df[df.cluster == 0]
df2 = df[df.cluster == 1]
df3 = df[df.cluster == 2]
df4 = df[df.cluster == 3]
```

In [70]:
```
plt.figure(figsize=(4,3))
plt.scatter(df1['Age'], df1['BMI'],color='y')

plt.scatter(df2['Age'], df2['BMI'],color='c')

plt.scatter(df3['Age'], df3['BMI'],color='r')

plt.scatter(df4['Age'], df4['BMI'],color='g')


plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color='k', marke
```

Out[70]: <matplotlib.collections.PathCollection at 0x123787d3390>



In [ ]:
```

```

In [71]:
```
import numpy as np
c =np.array([[1,2,3], [3,2,5]])
c[:,2]
```

Out[71]: array([3, 5])

In [72]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

df =pd.read_csv('salary.csv')
df
```

Out[72]:

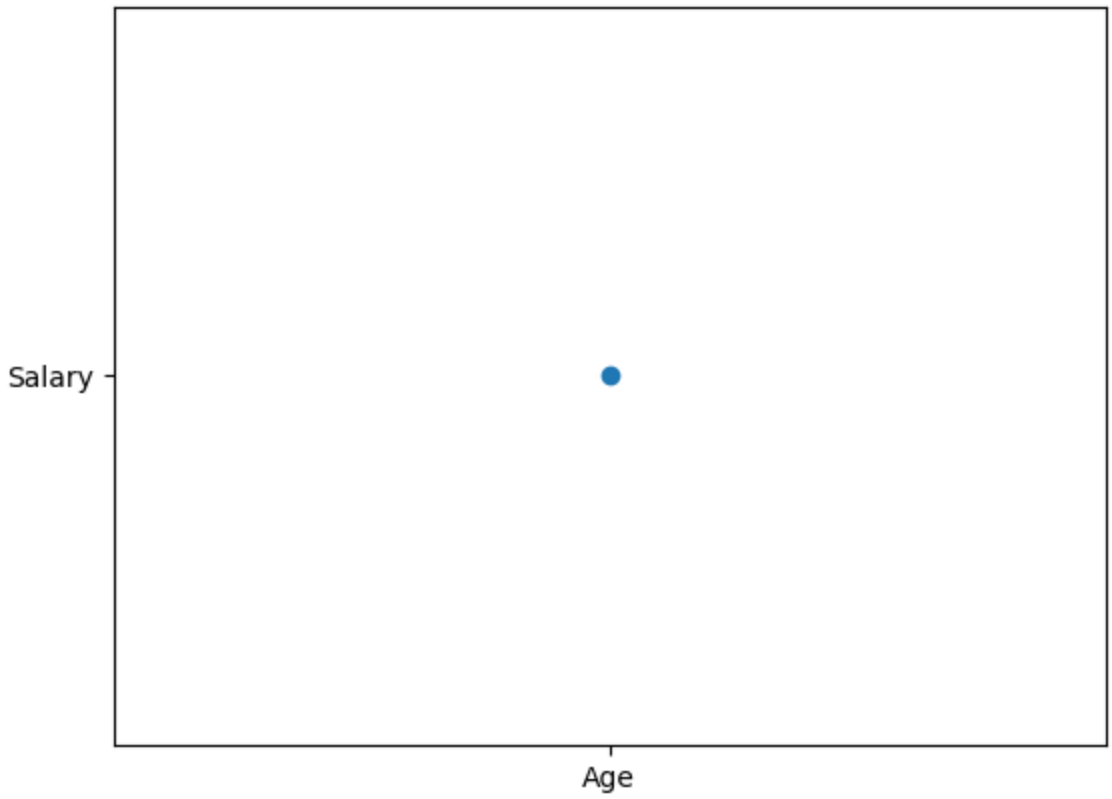
	Age	Salary
0	25	5562
1	28	4007
2	27	3083
3	30	3496
4	32	3298
5	38	4338
6	39	2513
7	40	4671
8	42	3816
9	43	4945
10	45	2672

In [73]:
```
df.sample()
```

Out[73]:

	Age	Salary
6	39	2513

In [74]:
```
plt.scatter(['Age'],['Salary'])
plt.show()
```



In [75]:
```
model = KMeans(n_clusters= 2)
```

In [76]:
```
df['cluster']= model.fit_predict(df[['Age','Salary']])
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_i
nit` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicit
ly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [77]:
```
df.cluster=model.fit_predict(df[['Age', 'Salary']])
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_i
nit` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicit
ly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [78]:
```
model.cluster_centers_
```

Out[78]:
```
array([[  34.71428571, 3269.28571429],
       [  36.5       , 4879.        ]])
```

In [79]:
```
model.cluster_centers_[:,0]
```

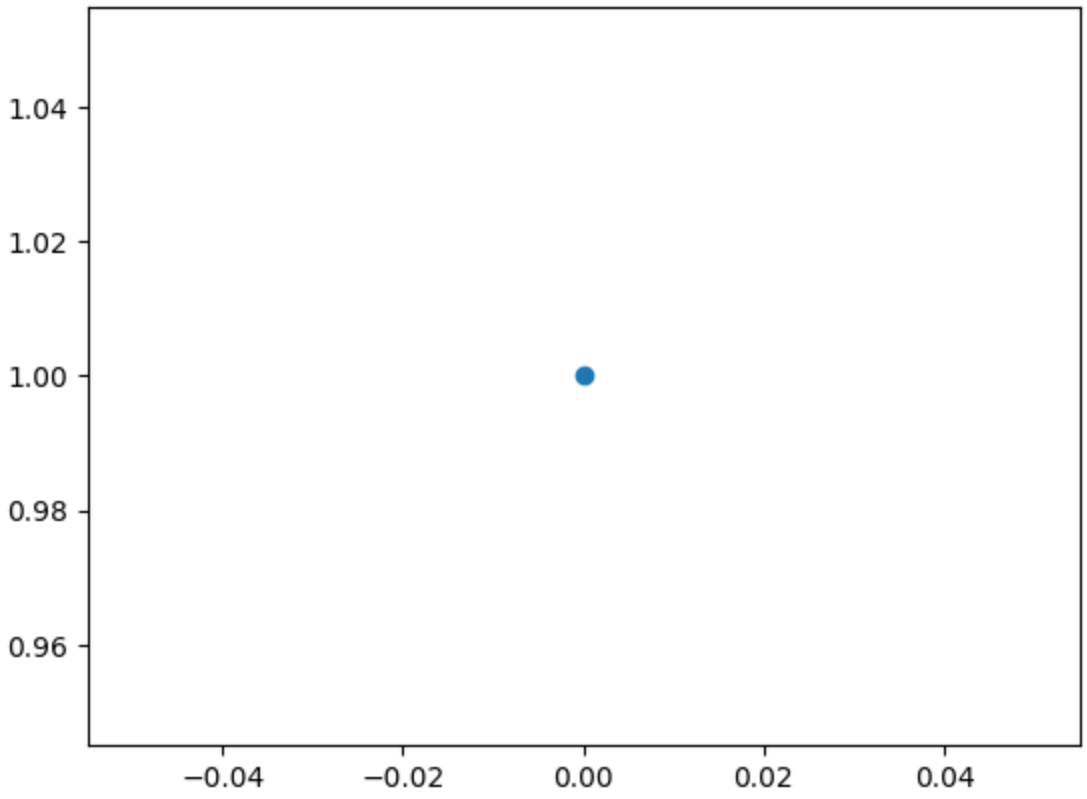
Out[79]:
```
array([34.71428571, 36.5       ])
```

In [80]:
```
model.cluster_centers_[:,1]
```

Out[80]:
```
array([3269.28571429, 4879.        ])
```

In [81]:
```
plt.scatter(0,1)
```

Out[81]:
```
<matplotlib.collections.PathCollection at 0x1237873d190>
```



In [82]:
```
df1 = df[df.cluster == 0]
df2 = df[df.cluster == 1]
```

In [83]:
```
plt.figure(figsize=(4,3))
plt.scatter(df1['Age'], df1['Salary'],color='y')
plt.scatter(df2['Age'], df2['Salary'],color='c')
```

Out[83]: <matplotlib.collections.PathCollection at 0x12379f6fb10>