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

In [2]:
```
df = pd.read_csv('HRdata1.csv')
```

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
Name                  0
Gender                0
Age                   0
10th                  0
12th                  0
Graduation            0
 Post Graduation      0
Experience            0
Salary                0
Performance Rating    0
ROI                   0
dtype: int64
```

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

Out[4]:

	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Perform Ra
count	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.00
mean	34.585714	64.585714	74.585714	54.585714	74.585714	4.585714	52928.571429	2.91
std	3.019228	3.019228	3.019228	3.019228	3.019228	3.019228	15096.137743	0.60
min	30.000000	60.000000	70.000000	50.000000	70.000000	0.000000	30000.000000	2.00
25%	32.000000	62.000000	72.000000	52.000000	72.000000	2.000000	40000.000000	2.40
50%	34.500000	64.500000	74.500000	54.500000	74.500000	4.500000	52500.000000	2.90
75%	37.000000	67.000000	77.000000	57.000000	77.000000	7.000000	65000.000000	3.40
max	40.000000	70.000000	80.000000	60.000000	80.000000	10.000000	80000.000000	4.00

In [5]:
```
#df.drop('Unnamed: 11', axis=1,inplace=True)
```

In [6]:
```
df.head(5)
```

Out[6]:

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating	
0	Aarti Panchal	Female	30	60	70	50	70	0	30000	2.0	2
1	Aastha Behl	Female	31	61	71	51	71	1	35000	2.2	2
2	Abhinaw Sinha	Male	32	62	72	52	72	2	40000	2.4	2

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating	
3	Abhishek Dabb	Male	33	63	73	53	73	3	45000	2.6	2
4	Abhishek Kumar Preetam	Male	34	64	74	54	74	4	50000	2.8	2

In [7]:
```
#df.drop([0, 49],inplace=True)
```

In [8]:
```
df.describe()
```

Out[8]:

	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Perform Ra
count	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.00
mean	34.585714	64.585714	74.585714	54.585714	74.585714	4.585714	52928.571429	2.91
std	3.019228	3.019228	3.019228	3.019228	3.019228	3.019228	15096.137743	0.60
min	30.000000	60.000000	70.000000	50.000000	70.000000	0.000000	30000.000000	2.00
25%	32.000000	62.000000	72.000000	52.000000	72.000000	2.000000	40000.000000	2.40
50%	34.500000	64.500000	74.500000	54.500000	74.500000	4.500000	52500.000000	2.90
75%	37.000000	67.000000	77.000000	57.000000	77.000000	7.000000	65000.000000	3.40
max	40.000000	70.000000	80.000000	60.000000	80.000000	10.000000	80000.000000	4.00

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

Out[9]:
```
(70, 11)
```

In [36]:
```
df.corr()
```

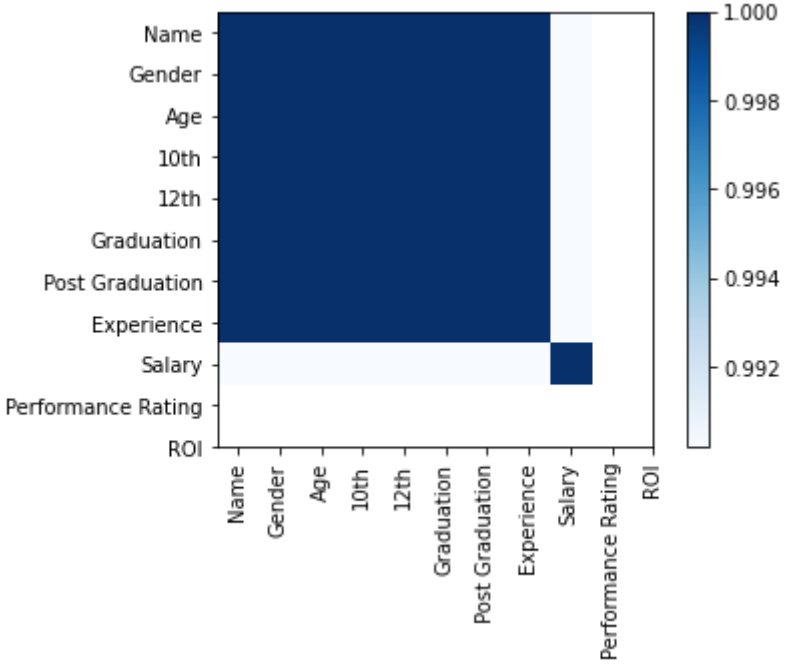
Out[36]:

	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performa Rat
Age	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
10th	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
12th	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
Graduation	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
Post Graduation	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
Experience	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
Salary	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
Performance Rating	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000

	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performa Rat
ROI	0.990199	0.990199	0.990199	0.990199	0.990199	0.990199	0.990199	0.990

In [37]:
```
plt.imshow(df.corr(), cmap=plt.cm.Blues, interpolation='nearest')
plt.colorbar()
tick_marks = [i for i in range(len(df.columns))]
plt.xticks(tick_marks, df.columns, rotation='vertical')
plt.yticks(tick_marks, df.columns)
```

Out[37]:
```
([<matplotlib.axis.YTick at 0x231317edd60>,
  <matplotlib.axis.YTick at 0x2313ebf5ac0>,
  <matplotlib.axis.YTick at 0x2313ebf2970>,
  <matplotlib.axis.YTick at 0x2313ec8f880>,
  <matplotlib.axis.YTick at 0x2313ec894f0>,
  <matplotlib.axis.YTick at 0x2313ec82520>,
  <matplotlib.axis.YTick at 0x2313ec8f730>,
  <matplotlib.axis.YTick at 0x2313ec97160>,
  <matplotlib.axis.YTick at 0x2313ec977f0>,
  <matplotlib.axis.YTick at 0x2313ec97f40>,
  <matplotlib.axis.YTick at 0x2313ec97f10>],
 [Text(0, 0, 'Name'),
  Text(0, 1, 'Gender'),
  Text(0, 2, 'Age'),
  Text(0, 3, '10th'),
  Text(0, 4, '12th'),
  Text(0, 5, 'Graduation'),
  Text(0, 6, ' Post Graduation'),
  Text(0, 7, 'Experience'),
  Text(0, 8, 'Salary'),
  Text(0, 9, 'Performance Rating'),
  Text(0, 10, 'ROI')])
```



In [10]:
```
df['Experience'] = df['Experience'].fillna(0)
```

In [11]:
```
df.head()
```

Out[11]:

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating	
0	Aarti Panchal	Female	30	60	70	50	70	0	30000	2.0	2
1	Aastha Behl	Female	31	61	71	51	71	1	35000	2.2	2
2	Abhinaw Sinha	Male	32	62	72	52	72	2	40000	2.4	2
3	Abhishek Dabb	Male	33	63	73	53	73	3	45000	2.6	2
4	Abhishek Kumar Preetam	Male	34	64	74	54	74	4	50000	2.8	2

In [12]:
```
df
```

Out[12]:

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating
0	Aarti Panchal	Female	30	60	70	50	70	0	30000	2.0
1	Aastha Behl	Female	31	61	71	51	71	1	35000	2.2
2	Abhinaw Sinha	Male	32	62	72	52	72	2	40000	2.4
3	Abhishek Dabb	Male	33	63	73	53	73	3	45000	2.6
4	Abhishek Kumar Preetam	Male	34	64	74	54	74	4	50000	2.8
...	...	...	...	...	...	...	...	...	...	...
65	Amery Ofer	Male	34	64	74	54	74	4	50000	2.8
66	Ameya Loke	Male	33	63	73	53	73	3	45000	2.6
67	Amii Elms	Male	32	62	72	52	72	2	40000	2.4
68	Amitie Mawson	Female	31	61	71	51	71	1	35000	2.2
69	Amol Nar	Male	30	60	70	50	70	0	30000	2.0

70 rows × 11 columns

In [13]:
```
df.describe()
```

Out[13]:

	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Perform Ra
count	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.00
mean	34.585714	64.585714	74.585714	54.585714	74.585714	4.585714	52928.571429	2.91
std	3.019228	3.019228	3.019228	3.019228	3.019228	3.019228	15096.137743	0.60
min	30.000000	60.000000	70.000000	50.000000	70.000000	0.000000	30000.000000	2.00
25%	32.000000	62.000000	72.000000	52.000000	72.000000	2.000000	40000.000000	2.40
50%	34.500000	64.500000	74.500000	54.500000	74.500000	4.500000	52500.000000	2.90
75%	37.000000	67.000000	77.000000	57.000000	77.000000	7.000000	65000.000000	3.40
max	40.000000	70.000000	80.000000	60.000000	80.000000	10.000000	80000.000000	4.00

In [14]:
```
df.max()
```

Out[14]:
```
Name                  Amol Nar
Gender                    Male
Age                         40
10th                        70
12th                        80
Graduation                  60
 Post Graduation            80
Experience                  10
Salary                   80000
Performance Rating         4.0
ROI                      35000
dtype: object
```

In [15]:
```
df['Salary'].max()
```

Out[15]:
```
80000
```

In [16]:
```
df['Salary'].mean()
```

Out[16]:
```
52928.57142857143
```

In [17]:
```
df['Performance Rating'].mean()
```

Out[17]:
```
2.917142857142857
```

In [18]:
```
df['ROI'].mean()
```

Out[18]:
```
29530.47142857143
```

In [19]:
```
df.rename(columns={"Graduation %": "Graduation", " Post Graduation %": "Post_Graduat
```

In [20]:
```
df.head(5)
```

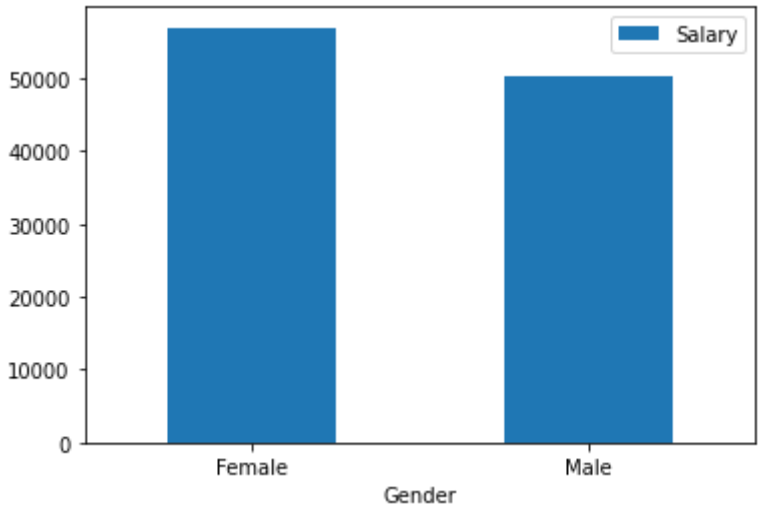
Out[20]:

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating	
0	Aarti Panchal	Female	30	60	70	50	70	0	30000	2.0	2
1	Aastha Behl	Female	31	61	71	51	71	1	35000	2.2	2
2	Abhinaw Sinha	Male	32	62	72	52	72	2	40000	2.4	2
3	Abhishek Dabb	Male	33	63	73	53	73	3	45000	2.6	2
4	Abhishek Kumar Preetam	Male	34	64	74	54	74	4	50000	2.8	2

In [21]:
```
y=df.groupby("Gender").mean()
```

In [22]:
```
y.plot.bar( y='Salary', rot=0)
```

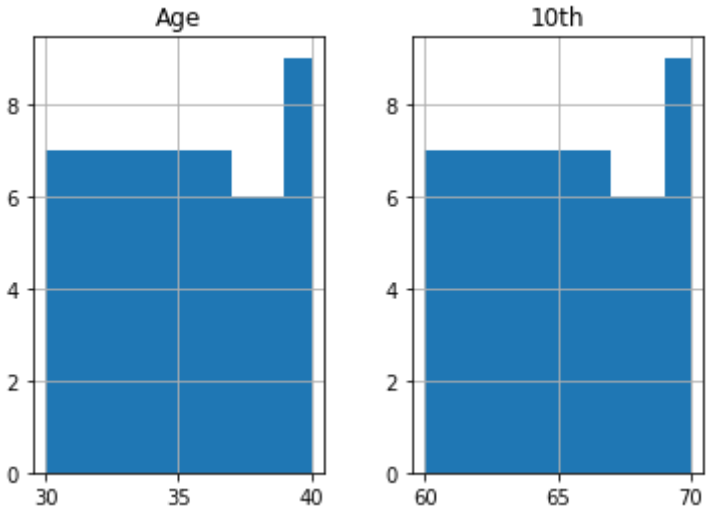
Out[22]: <AxesSubplot:xlabel='Gender'>



In [23]:
```
df.iloc[:, 2:4].hist()
```

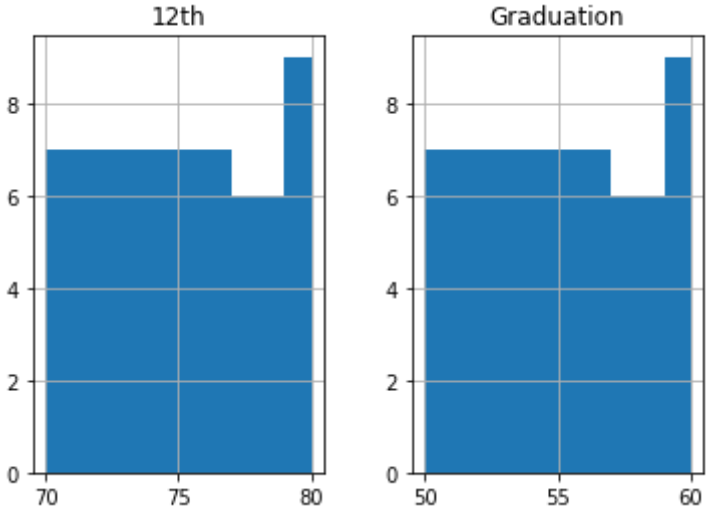
```
c:\users\prashant\appdata\local\programs\python\python39\lib\site-packages\pandas\pl
otting\_matplotlib\tools.py:400: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed two m
inor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```

Out[23]:
```
array([[<AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'10th'}>]], dtype=object)
```



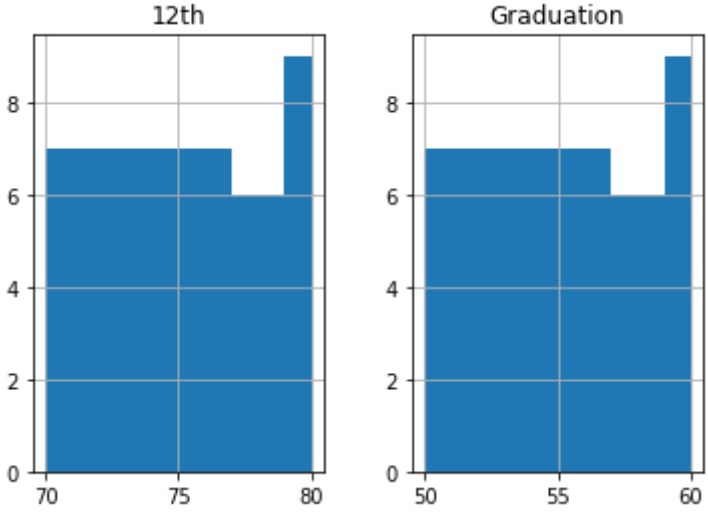
In [24]:
```
df.iloc[:, 4:6].hist()
```

Out[24]:
```
array([[<AxesSubplot:title={'center':'12th'}>,
        <AxesSubplot:title={'center':'Graduation'}>]], dtype=object)
```



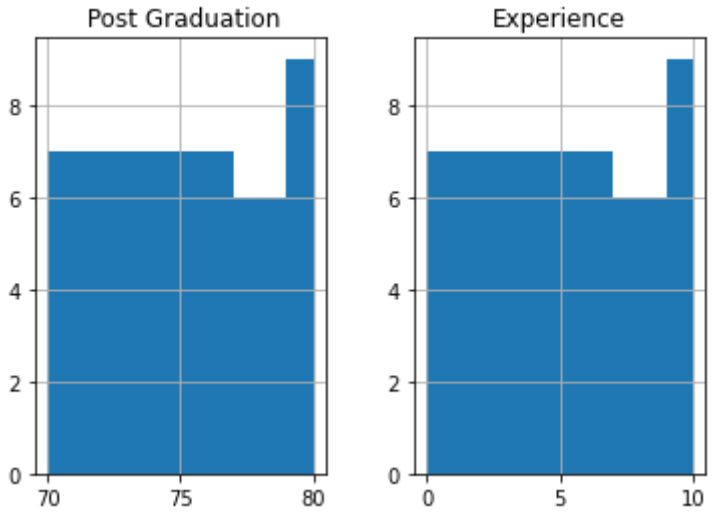
In [25]:
```
df.iloc[:, 4:6].hist()
```

Out[25]:
```
array([[<AxesSubplot:title={'center':'12th'}>,
        <AxesSubplot:title={'center':'Graduation'}>]], dtype=object)
```



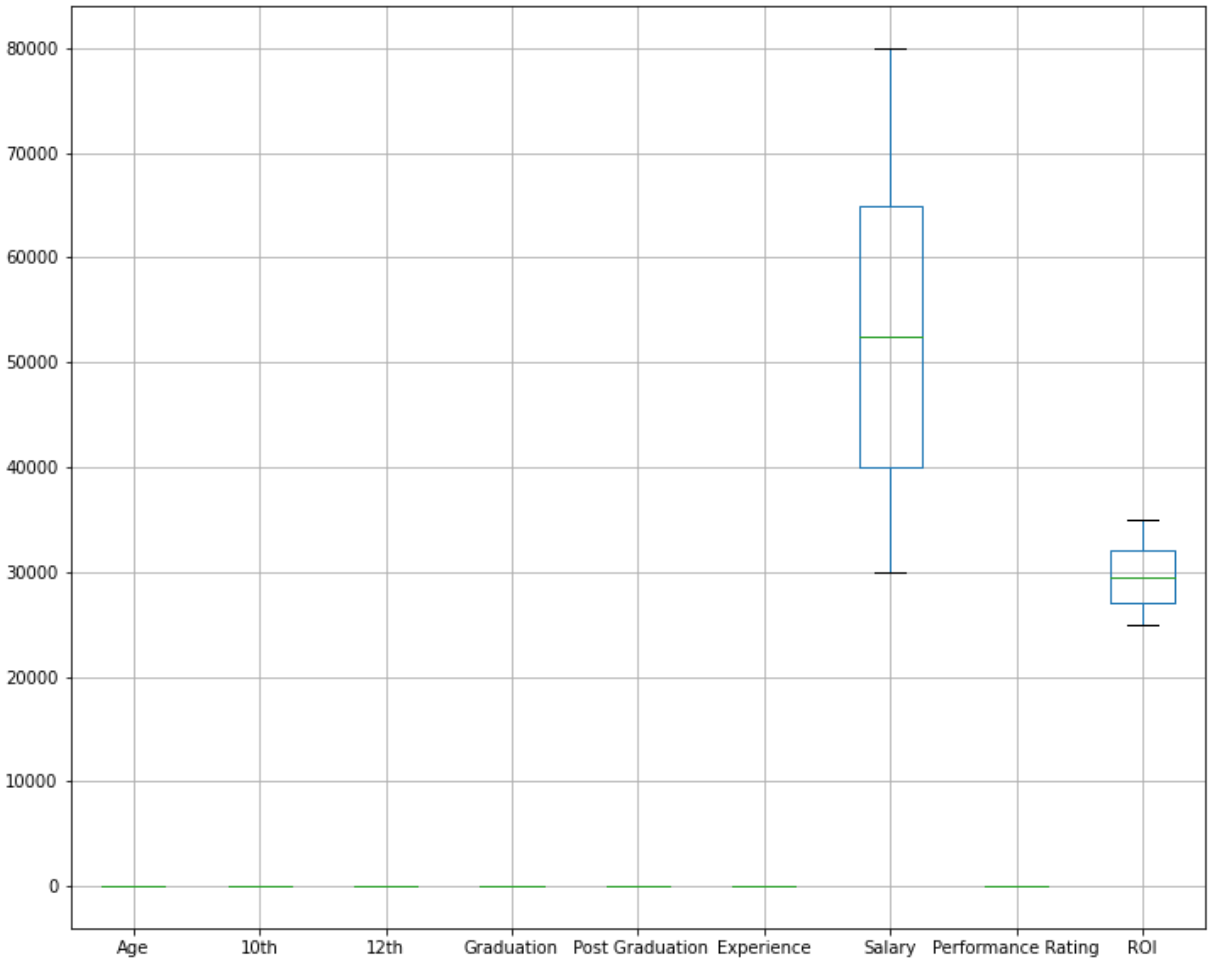
In [26]:
```
df.iloc[:, 6:8].hist()
```

Out[26]:
```
array([[<AxesSubplot:title={'center':' Post Graduation'}>,
        <AxesSubplot:title={'center':'Experience'}>]], dtype=object)
```



In [27]:
```
df.boxplot(figsize=(12,10))
```

Out[27]:
```
<AxesSubplot:>
```



In [28]:
```
df.head(1)
```

Out[28]:

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating	
0	Aarti Panchal	Female	30	60	70	50	70	0	30000	2.0	25

In [29]:
```
#setting features
X = df.iloc[:,2:8].values
y = df.iloc[:,9].values
```

In [30]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_st
X_test[0]
df.head(1)
```

Out[30]:

	Name	Gender	Age	10th	12th	Graduation	Post Graduation	Experience	Salary	Performance Rating	
0	Aarti Panchal	Female	30	60	70	50	70	0	30000	2.0	25

In [31]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
z=[[40, 70, 80, 40, 70, 10]]
# Predicting the Test set results
y_pred = regressor.predict(X_test)
#y_pred = regressor.predict(z)
```

In [32]:
```
from sklearn.metrics import r2_score,mean_squared_error
mse = mean_squared_error(y_test,y_pred)
rsq = r2_score(y_test,y_pred)
print('mean squared error :',mse)
print('r square :',rsq)
```

```
mean squared error : 8.170345089789052e-31
r square : 1.0
```

In [33]:
```
import pickle
print(pickle.__doc__)
pickle.dump(regressor, open("ml_model.sav", "wb"))
```

```
Create portable serialized representations of Python objects.

See module copyreg for a mechanism for registering custom picklers.
See module pickletools source for extensive comments.

Classes:

    Pickler
    Unpickler

Functions:

    dump(object, file)
    dumps(object) -> string
    load(file) -> object
    loads(bytes) -> object

Misc variables:

    __version__
    format_version
```

```
    compatible_formats
```

In [34]:
```
rf = pickle.load(open('ml_model.sav','rb'))
z=[[40, 70, 80, 0, 70, 0]]
prediction = rf.predict(z)
```

In [35]:
```
prediction[0]
```

Out[35]:
```
1.476254468717391
```