Importing Reqired Libraries

In [57]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
import warnings
warnings.filterwarnings('ignore')
```

Import Data

In [6]:
```
election = pd.read_csv("election_data.csv")
election
```

Out[6]:

	Election-id	Result	Year	Amount Spent	Popularity Rank
0	NaN	NaN	NaN	NaN	NaN
1	122.0	0.0	32.0	3.81	3.0
2	315.0	1.0	48.0	6.32	2.0
3	201.0	1.0	51.0	3.67	1.0
4	965.0	0.0	40.0	2.93	4.0
5	410.0	1.0	52.0	3.60	1.0
6	150.0	0.0	35.0	4.20	4.0
7	743.0	1.0	39.0	5.66	2.0
8	612.0	1.0	42.0	4.32	3.0
9	206.0	1.0	44.0	3.26	3.0
10	792.0	0.0	50.0	4.52	4.0

Initial Analysis

In [7]:
```
election.shape
```

Out[7]:
```
(11, 5)
```

In [8]:
```
election.dtypes
```

Out[8]:
```
Election-id        float64
Result             float64
Year               float64
Amount Spent       float64
```

```
Popularity Rank    float64
dtype: object
```

In [9]:
```
election.isna().sum()
```

Out[9]:
```
Election-id        1
Result             1
Year               1
Amount Spent       1
Popularity Rank    1
dtype: int64
```

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

Out[10]:

	Election-id	Result	Year	Amount Spent	Popularity Rank
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	451.600000	0.600000	43.300000	4.229000	2.700000
std	304.266404	0.516398	6.912951	1.055351	1.159502
min	122.000000	0.000000	32.000000	2.930000	1.000000
25%	202.250000	0.000000	39.250000	3.617500	2.000000
50%	362.500000	1.000000	43.000000	4.005000	3.000000
75%	710.250000	1.000000	49.500000	4.470000	3.750000
max	965.000000	1.000000	52.000000	6.320000	4.000000

In [11]:
```
election.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Election-id      10 non-null     float64
 1   Result           10 non-null     float64
 2   Year             10 non-null     float64
 3   Amount Spent     10 non-null     float64
 4   Popularity Rank  10 non-null     float64
dtypes: float64(5)
memory usage: 568.0 bytes
```

In [12]:
```
election.dropna(axis = 0,inplace = True)
```

In [15]:
```
election.head(10)
```

Out[15]:

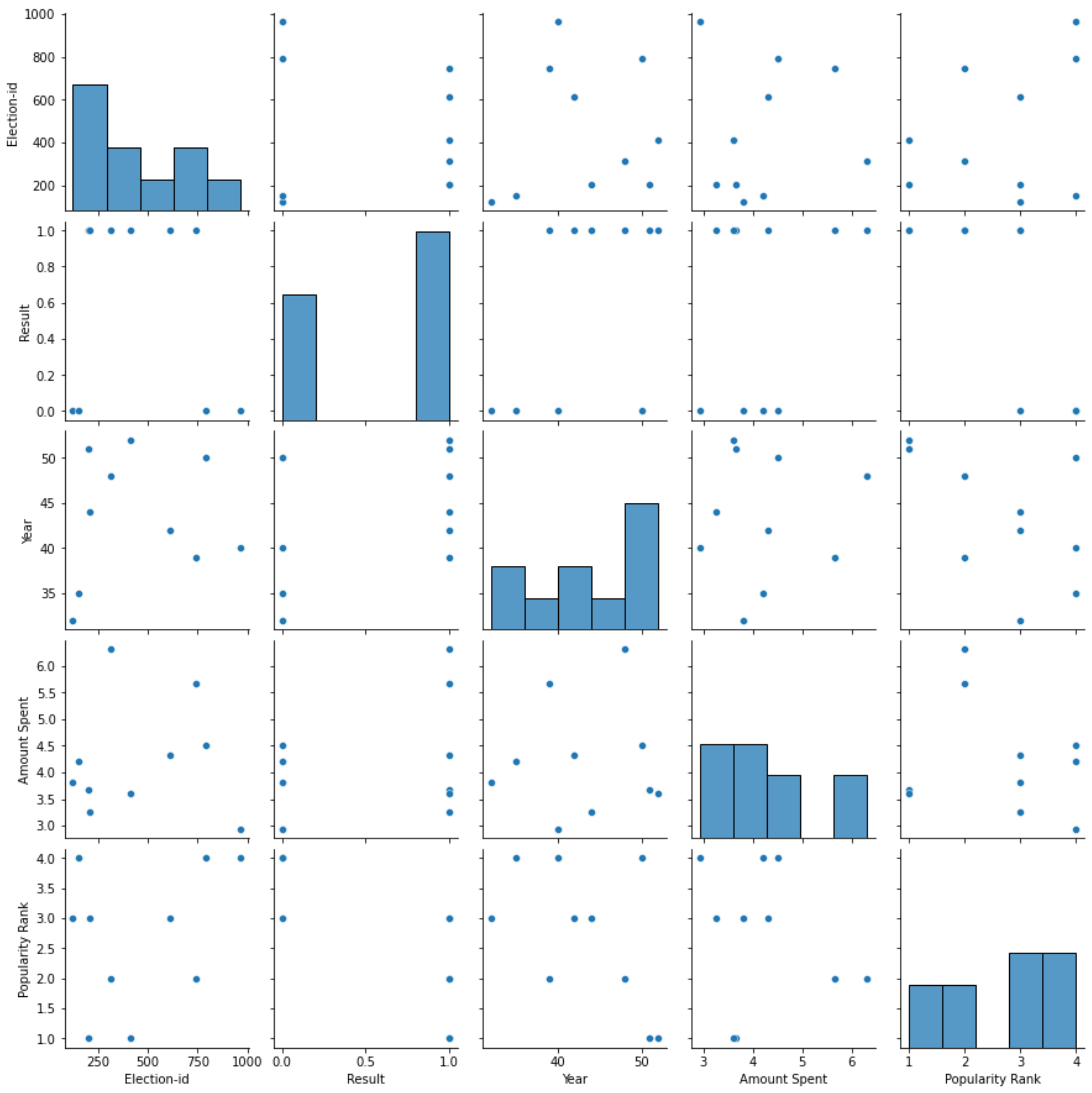
	Election-id	Result	Year	Amount Spent	Popularity Rank
1	122.0	0.0	32.0	3.81	3.0
2	315.0	1.0	48.0	6.32	2.0
3	201.0	1.0	51.0	3.67	1.0
4	965.0	0.0	40.0	2.93	4.0
5	410.0	1.0	52.0	3.60	1.0

	Election-id	Result	Year	Amount Spent	Popularity Rank
6	150.0	0.0	35.0	4.20	4.0
7	743.0	1.0	39.0	5.66	2.0
8	612.0	1.0	42.0	4.32	3.0
9	206.0	1.0	44.0	3.26	3.0
10	792.0	0.0	50.0	4.52	4.0

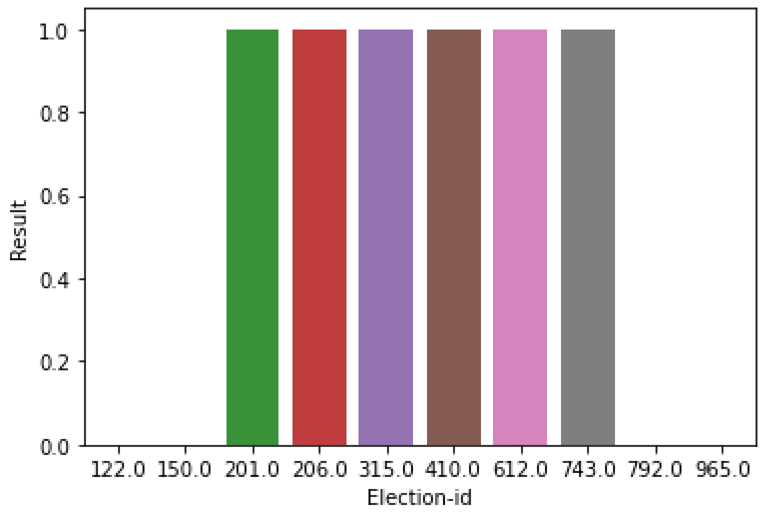
In [16]:
```
election.shape
```

Out[16]:
```
(10, 5)
```

In [19]:
```
sns.pairplot(election)
plt.show()
```

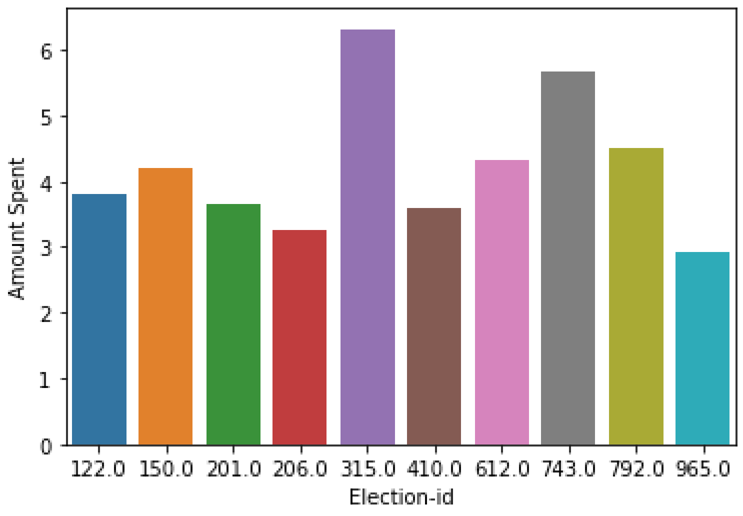


In [30]:
```
sns.barplot(x = 'Election-id' , y = 'Result' ,data = election)
plt.show()
```



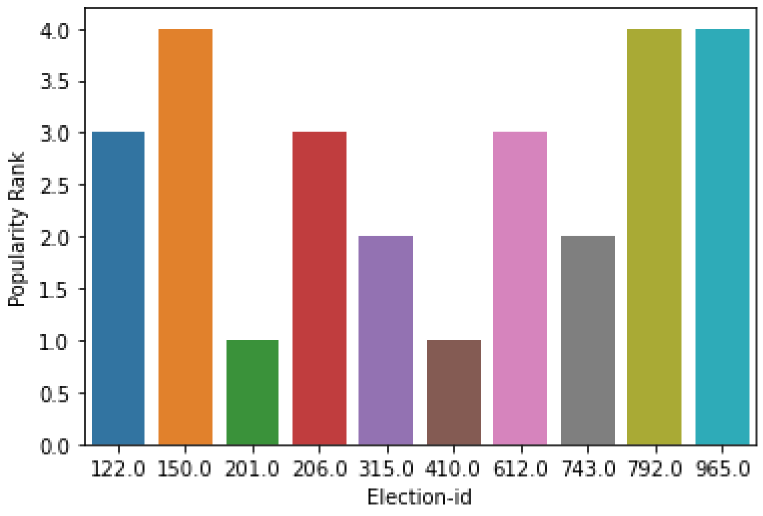
In [33]:

```
sns.barplot( x = 'Election-id', y = 'Amount Spent' ,data = election)
plt.show()
```



In [36]:

```
sns.barplot(x = 'Election-id' , y = 'Popularity Rank' ,data = election)
plt.show()
```



Model Building

In [37]:

```
x = election.iloc[:,1:]
y = election.iloc[:,0]
```

In [39]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.20)
```

In [40]:
```
x_train.shape,y_train.shape
```

Out[40]:
```
((8, 4), (8,))
```

In [41]:
```
x_test.shape,y_train.shape
```

Out[41]:
```
((2, 4), (8,))
```

Model Training

In [45]:
```
classifier = LogisticRegression()
classifier.fit(x_train,y_train)
```

Out[45]:
```
LogisticRegression()
```

In [73]:
```
y_train_pred  = classifier.predict(x_train)
y_train_pred
```

Out[73]:
```
array([315., 201., 965., 612., 792., 206., 743., 150.])
```

In [74]:
```
accuracy_score(y_train,y_train_pred)
```

Out[74]:
```
1.0
```

In [75]:
```
confusion_matrix(y_train,y_train_pred)
```

Out[75]:
```
array([[1, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 1]], dtype=int64)
```

In [76]:
```
print(classification_report(y_train,y_train_pred))
```

```
              precision    recall  f1-score   support

       150.0       1.00      1.00      1.00         1
       201.0       1.00      1.00      1.00         1
       206.0       1.00      1.00      1.00         1
       315.0       1.00      1.00      1.00         1
       612.0       1.00      1.00      1.00         1
       743.0       1.00      1.00      1.00         1
       792.0       1.00      1.00      1.00         1
       965.0       1.00      1.00      1.00         1
```

```
    accuracy                           1.00         8
   macro avg       1.00      1.00      1.00         8
weighted avg       1.00      1.00      1.00         8
```

In [77]:

```
y_pred_df= pd.DataFrame({'actual': y,
                         'predicted_prob': classifier.predict(x)})
y_pred_df
```

Out[77]:

	actual	predicted_prob
1	122.0	150.0
2	315.0	315.0
3	201.0	201.0
4	965.0	965.0
5	410.0	201.0
6	150.0	150.0
7	743.0	743.0
8	612.0	612.0
9	206.0	206.0
10	792.0	792.0

In [78]:

```
classifier.predict_proba (x)[:,1]
```

Out[78]:

```
array([8.98847989e-05, 8.42757336e-02, 7.52111542e-01, 9.27710323e-03,
       7.71635804e-01, 3.47819023e-04, 7.25505460e-03, 4.80038441e-02,
       1.19117333e-01, 5.48228252e-02])
```

In [95]:

```
### roc curve



fpr, tpr, thresholds = roc_curve(y_train,classifier.predict_proba (x)[:,1])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-95-1126a337eff0> in <module>
      3 
      4 
----> 5 fpr, tpr, thresholds = roc_curve(y_train,classifier.predict_proba (x)[:,1])
      6 
      7 

~\anaconda3.01\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwa
rgs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

~\anaconda3.01\lib\site-packages\sklearn\metrics\_ranking.py in roc_curve(y_true, y_
score, pos_label, sample_weight, drop_intermediate)
    911 
    912     """
--> 913     fps, tps, thresholds = _binary_clf_curve(
    914         y_true, y_score, pos_label=pos_label, sample_weight=sample_weight)
    915 

~\anaconda3.01\lib\site-packages\sklearn\metrics\_ranking.py in _binary_clf_curve(y_
true, y_score, pos_label, sample_weight)
    689     if not (y_type == "binary" or
    690             (y_type == "multiclass" and pos_label is not None)):
--> 691         raise ValueError("{0} format is not supported".format(y_type))
    692 
    693     check_consistent_length(y_true, y_score, sample_weight)

ValueError: multiclass format is not supported
```

```
In [ ]:
```