In [4]:
```
### START OF PROJECT ###
```

In [5]:
```
#import necessary libraries
import numpy as np
import pandas as pd
```

In [6]:
```
#retrive dataset information
df=pd.read_csv('mini.csv')
df
```

Out[6]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	R
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	
1	NaN	Stock control & order processing;	New Development	Multi	4GL	
2	NaN	Billing;	Enhancement	NaN	3GL	
3	Client server	NaN	Enhancement	NaN	NaN	
4	Client server	Management Information System;	Enhancement	MF	3GL	
...	...	...	...	...	...	
6755	NaN	NaN	New Development	NaN	NaN	
6756	Stand alone	Management Information System;	Enhancement	MF	3GL	
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	
6759	Client server	Cars selling;	Enhancement	Multi	3GL	

6760 rows × 15 columns

In [8]:
```
# check particular column frequent occurance
df['Architecture'].value_counts()
```

Out[8]:
```
Client server                          1371
Stand alone                            1295
Multi-tier                              382
Multi-tier / Client server              275
Multi-tier with web public interface    164
Multi-tier with web interface            25
Stand-alone                              14
Name: Architecture, dtype: int64
```

In [9]:
```
# consider the most frequent occurance
df['ApplicationType'].value_counts()
```

Out[9]:
```
Financial transaction process/accounting;
987
Transaction/Production System;
497
not recorded;
477
Management Information System;
375
relatively complex application;
154

...
GPS Portal;
1
Airport Management;
1
Trading;Electronic Data Interchange;
1
Office Information System;Case Management System;
1
Decision Support System;Management Information System;Office Information Syste
m;      1
Name: ApplicationType, Length: 556, dtype: int64
```

In [10]:
```
#after preprocessing with frequent filling pattern
df = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
df
```

Out[10]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	R
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	
1	Client server	Stock control & order processing;	New Development	Multi	4GL	
2	Client server	Billing;	Enhancement	MF	3GL	
3	Client server	Financial transaction process/accounting;	Enhancement	MF	3GL	
4	Client server	Management Information System;	Enhancement	MF	3GL	
...	...	...	...	...	...	
6755	Client server	Financial transaction process/accounting;	New Development	MF	3GL	
6756	Stand alone	Management Information System;	Enhancement	MF	3GL	
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	
6759	Client server	Cars selling;	Enhancement	Multi	3GL	

6760 rows × 15 columns

In [7]:
```
# checking for any missing data
df.isnull().sum()
```

Out[7]:
```
Architecture            0
ApplicationType         0
DevelopmentType         0
DevelopmentPlatform     0
LanguageType            0
RelativeSize            0
UsedMethodology         0
AgileMethodUsed         0
ResourceLevel           0
PackageCustomisation    0
IndustrySector          0
Effort                  0
Projectelaspedtime      0
Projectinactivetime     0
Duration                0
dtype: int64
```

In [11]:
```
#dropping unwanted columns
data=df.drop(['Projectelaspedtime','Projectinactivetime'],axis=1)
data
```

Out[11]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	R
0	Stand alone	Transaction/Production System;	New Development	MR	4GL	
1	Client server	Stock control & order processing;	New Development	Multi	4GL	
2	Client server	Billing;	Enhancement	MF	3GL	
3	Client server	Financial transaction process/accounting;	Enhancement	MF	3GL	
4	Client server	Management Information System;	Enhancement	MF	3GL	
...	...	...	...	...	...	
6755	Client server	Financial transaction process/accounting;	New Development	MF	3GL	
6756	Stand alone	Management Information System;	Enhancement	MF	3GL	
6757	Multi-tier	Customer relationship management;	Enhancement	Multi	3GL	
6758	Stand alone	Electronic Data Interchange;	New Development	PC	3GL	
6759	Client server	Cars selling;	Enhancement	Multi	3GL	

6760 rows × 13 columns

In [12]:
```
# label encoding the data
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Architecture']= le.fit_transform(data['Architecture'])
data['ApplicationType']= le.fit_transform(data['ApplicationType'])
data['DevelopmentType']= le.fit_transform(data['DevelopmentType'])
data['DevelopmentPlatform']= le.fit_transform(data['DevelopmentPlatform'])
data['LanguageType']= le.fit_transform(data['LanguageType'])
data['RelativeSize']= le.fit_transform(data['RelativeSize'])
data['UsedMethodology']= le.fit_transform(data['UsedMethodology'])
data['AgileMethodUsed']= le.fit_transform(data['AgileMethodUsed'])
data['ResourceLevel']= le.fit_transform(data['ResourceLevel'])
data['PackageCustomisation']= le.fit_transform(data['PackageCustomisation'])
data['IndustrySector']= le.fit_transform(data['IndustrySector'])
```

In [13]:
```
#after encoding
data
```

Out[13]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	Relativ
0	5	514	1	2	2	
1	0	474	1	3	2	
2	0	27	0	1	1	
3	0	209	0	1	1	
4	0	307	0	1	1	
...	...	...	...	...	...	
6755	0	209	1	1	1	
6756	5	307	0	1	1	
6757	1	137	0	3	1	
6758	5	183	1	4	1	
6759	0	61	0	3	1	

6760 rows × 13 columns

In [14]:
```
#converting alldata into binary format
binary_df = (data > 0).astype(int)
binary_df
```

Out[14]:

	Architecture	ApplicationType	DevelopmentType	DevelopmentPlatform	LanguageType	Relativ
0	1	1	1	1	1	
1	0	1	1	1	1	
2	0	1	0	1	1	
3	0	1	0	1	1	
4	0	1	0	1	1	
...	...	...	...	...	...	...
6755	0	1	1	1	1	
6756	1	1	0	1	1	
6757	1	1	0	1	1	
6758	1	1	1	1	1	
6759	0	1	0	1	1	

6760 rows × 13 columns

In [12]:
```
# mean calculation for effort
x=data['Effort'].mean()
print("Effort mean is : ",x)
```

```
Effort mean is :  5005.31124260355
```

In [13]:
```
# standard deviation calculation for effort
y=data['Effort'].std()
print("Effort standardDeviation is : ",y)
```

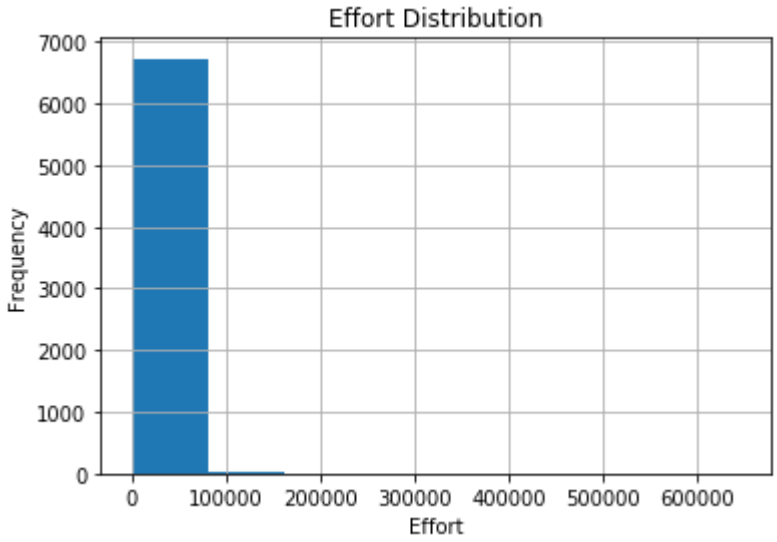
```
Effort standardDeviation is :  16773.124532616057
```

In [14]:
```
#normal distribution calculation for effort
import scipy.stats
z=scipy.stats.norm(5005.31, 16773.12).pdf(98)
print("Effort normal distribution is : ",z)
```

```
Effort normal distribution is :  2.278814781804906e-05
```

In [34]:
```
#plotting effort data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.figure(figsize=[5,8])
data.hist(column='Effort',bins=8)
plt.xlabel('Effort')
plt.ylabel('Frequency')
plt.title('Effort Distribution')
plt.show()
```

```
<Figure size 360x576 with 0 Axes>
```



In [16]:
```
# mean calculation for duration
w=data['Duration'].mean()
print("Duration mean is : ",w)
```

```
Duration mean is :  7.048316568047344
```

In [17]:
```
# standard deviation calculation for duration
m=data['Duration'].std()
print("Duration standard deviation is : ",m)
```

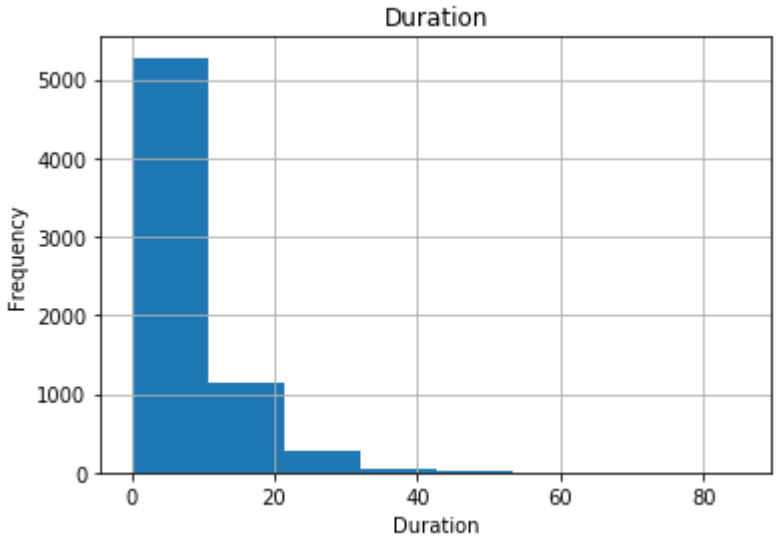
```
Duration standard deviation is :  6.978858870981555
```

In [18]:
```
#normal distribution calculation for duration
import scipy.stats
n=scipy.stats.norm(7.04, 6.97).pdf(100)
print("Duration normal distribution is : ",n)
```

```
Duration normal distribution is :  1.3538352346476263e-40
```

In [19]:
```
#plotting Duration data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.figure(figsize=[10,8])
data.hist(column='Duration',bins=8)
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.title('Duration')
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



In [20]:
```
#splitting into trainin and test data
X = binary_df.iloc[:, 1:12].values
y = binary_df.iloc[:, 12:14].values
```

In [21]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,rando
```

In [22]:
```
#fitting classifier
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
```

```
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarn
ing: The default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[22]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [23]:
```
y_pred = svclassifier.predict(X_test)
```

In [24]:
```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score

print("Accuracy of svm : ",accuracy_score(y_test, y_pred))

print("confusion matrix \n",
      confusion_matrix (y_test, y_pred))
```

```
Accuracy of svm :  0.8594674556213018
confusion matrix
 [[   0  190]
 [   0 1162]]
```

In [25]:
```
print("classification report \n")
print(classification_report(y_test, y_pred))
```

```
classification report

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       190
           1       0.86      1.00      0.92      1162

    accuracy                           0.86      1352
   macro avg       0.43      0.50      0.46      1352
weighted avg       0.74      0.86      0.79      1352


C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1
437: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [26]:
```
#regression metrics implementation
from sklearn.metrics import mean_absolute_error
mae=mean_absolute_error(y_test, y_pred)
print("MAE is %.2f " %mae)
```

```
MAE is 0.14
```

In [27]:
```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test, y_pred)
print("MSE is %.2f"%mse)
```

```
MSE is 0.14
```

In [28]:
```
from sklearn.metrics import max_error
me=max_error(y_test,y_pred)
print("ME is %.2f"%me)
```

```
ME is 1.00
```

In [29]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse = sqrt(mean_squared_error(y_test, y_pred))
print("RMSE is %.2f" %rmse)
```

```
RMSE is 0.37
```

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 = 0.20,rando
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(5408, 11)
(1352, 11)
(5408, 1)
(1352, 1)
```

In [59]:
```
#cross validation
X = binary_df.iloc[:, 1:12].values
y = binary_df.iloc[:, 12:14].values
```

In [60]:
```
from sklearn.model_selection import cross_validate
from sklearn.metrics import make_scorer

def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]
def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]
def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]
def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]
def acc(y_true, y_pred): return round(accuracy_score(y_true, y_pred),3)

#cross validation purpose
scoring = {'accuracy': make_scorer(accuracy_score),'prec': 'precision'}
scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
           'fp': make_scorer(fp), 'fn': make_scorer(fn),'accuracy': make_scorer(

def display_result(result):
    print("TP: ",result['test_tp'])
    print("TN: ",result['test_tn'])
    print("FN: ",result['test_fn'])
    print("FP: ",result['test_fp'])
    print("Accuracy : ",result['test_accuracy'])
```

In [61]:
```
result=cross_validate(clf,X_train,y_train,scoring=scoring,cv=3)
display_result(result)
```

```
TP:  [1578 1577 1577]
TN:  [0 0 0]
FN:  [0 0 0]
FP:  [226 225 225]
Accuracy :  [0.875 0.875 0.875]

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [42]:
```
#Logistic regression implementation
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix

clf=LogisticRegression()
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
print("Accuracy is : ",accuracy_score(y_test,y_pred))
print("confusion matrix \n")
print(confusion_matrix(y_test,y_pred))
```

```
Accuracy is :  0.8579881656804734
confusion matrix

[[   0  192]
 [   0 1160]]

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [43]:
```
#after cross validation
result=cross_validate(clf,X_train,y_train,scoring=scoring,cv=10)
display_result(result)
```

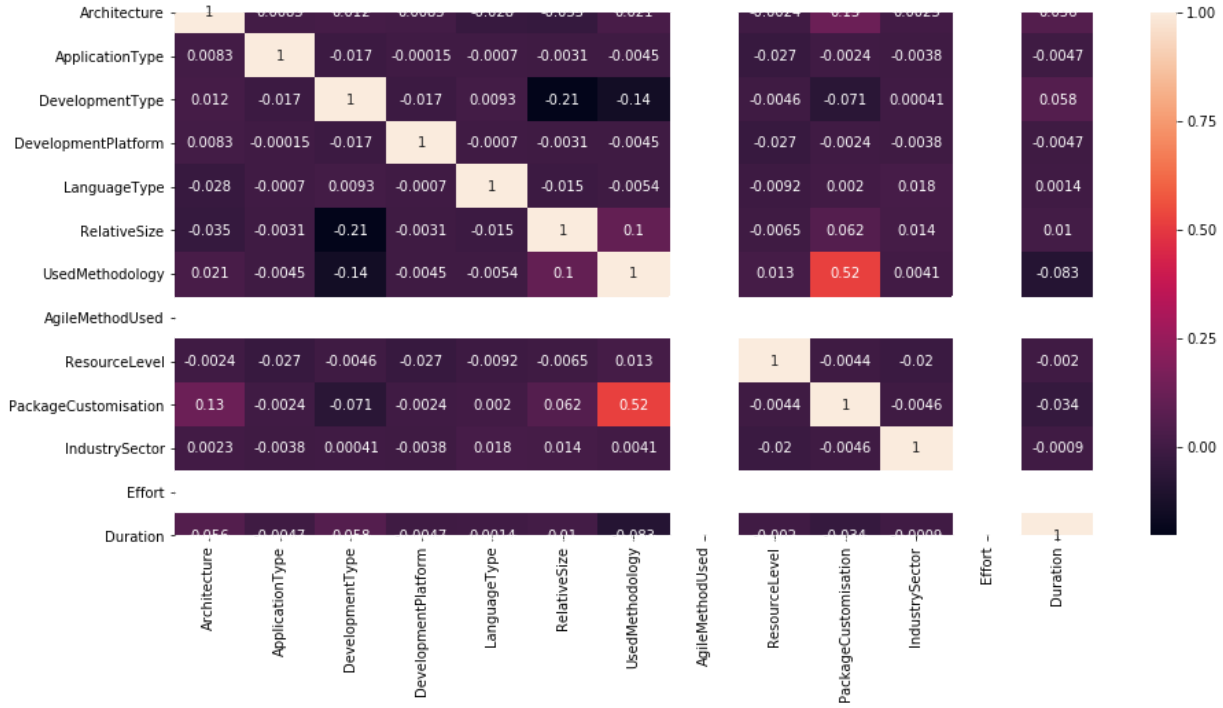
```
TP:  [474 474 473 473 473 473 473 473 473 473]
TN:  [0 0 0 0 0 0 0 0 0 0]
FN:  [0 0 0 0 0 0 0 0 0 0]
FP:  [68 68 68 68 68 68 67 67 67 67]
Accuracy :  [0.875 0.875 0.874 0.874 0.874 0.874 0.876 0.876 0.876 0.876]

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
```

```
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724: Da
taConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [44]:
```
#heat map representation
import seaborn as sns
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(binary_df.corr(), annot=True)
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x206056b2d08>



In [45]:
```
# calculating effort and duration
import pandas as pd
```

In [46]:
```
df=pd.read_csv('loc.csv')
df
```

Out[46]:

	LOC
0	1,250
1	47,250
2	6,800
3	5,200
4	1,516
...	...
517	33,500
518	1,760
519	10,080
520	453,824
521	300,000

522 rows × 1 columns

In [47]:
```
#convert to numeric
df['LOC'] = pd.to_numeric(df['LOC'], errors='coerce')
df
```

Out[47]:

	LOC
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
...	...
517	NaN
518	NaN
519	NaN
520	NaN
521	NaN

522 rows × 1 columns

In [48]:
```
df.describe()
```

Out[48]:

	LOC
count	29.000000
mean	469.724138
std	277.166775
min	96.000000
25%	215.000000
50%	410.000000
75%	702.000000
max	973.000000

In [49]:
```
#calculate mean for LOC
x=df['LOC'].mean()
print(int(x))
```

```
469
```

In [50]:
```
# effort calculation in person months
a=3.2
b=1.05
KLOC=469
effort=int(a*(KLOC)**b)
print("Effort","=",effort ,"(Person Months)")
```

```
Effort = 2041 (Person Months)
```

In [51]:
```
#duration calculation in months
c=2.5
d=0.38
Effort=2041
Duration = int(c*(Effort)**d)
print("Duration","=",Duration ,"(Months)")
```

```
Duration = 45 (Months)
```

In [52]:
```
#persons required to complete project
effort=2041
duration=45
PersonsRequired=effort//duration
print("Persons Required", "=", PersonsRequired)
```

```
Persons Required = 45
```

In [53]:
```
# effort calculation in person months
a=3
b=1.12
KLOC=469
effort=int(a*(KLOC)**b)
print("Effort","=",effort ,"(Person Months)")
```

```
Effort = 2943 (Person Months)
```

In [54]:
```
#duration calculation in months
c=2.5
d=0.35
Effort=2943
Duration = int(c*(Effort)**d)
print("Duration","=",Duration ,"(Months)")
```

```
Duration = 40 (Months)
```

In [55]:
```
#persons required to complete project
effort=2943
duration=40
PersonsRequired=effort//duration
print("Persons Required", "=", PersonsRequired)
```

```
Persons Required = 73
```

In [56]:
```
# effort calculation in person months
a=2.8
b=1.20
KLOC=469
effort=int(a*(KLOC)**b)
print("Effort","=",effort ,"(Person Months)")
```

```
Effort = 4493 (Person Months)
```

In [57]:
```
#duration calculation in months
c=2.5
d=0.32
Effort=4493
Duration = int(c*(Effort)**d)
print("Duration","=",Duration ,"(Months)")
```

```
Duration = 36 (Months)
```

In [58]:
```
#persons required to complete project
effort=4493
duration=36
PersonsRequired=effort//duration
print("Persons Required", "=", PersonsRequired)
```

```
Persons Required = 124
```

In [ ]:
```
### END OF PROJECT ###
```

In [ ]:
```

```