LOAN PREDICTION

CREATED BY SANDEEP GUPTA

Task 1: Importing libraries and Exploring the Dataset.

Task 2: Definining Exploratory Data Analysis with an overview of the whole project .

Task 3: Checking missing values and Outliers & Creating visual methods to analyze the data.

Task 4: creat a model that fits the data

Task 5: creating an accurAcy table

We are going to create an accuracy table for each model separately

TASK 1: IMPORTING LIBRARIES AND EXPLORING THE DATASET.

In [605]:

```
# Importante Required Libraries
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
import joblib
import pickle
```

In [607]:
```python
# Read The Dataset
data = pd.read_csv('loan_data.csv')
data.head()
```

Out[607]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0

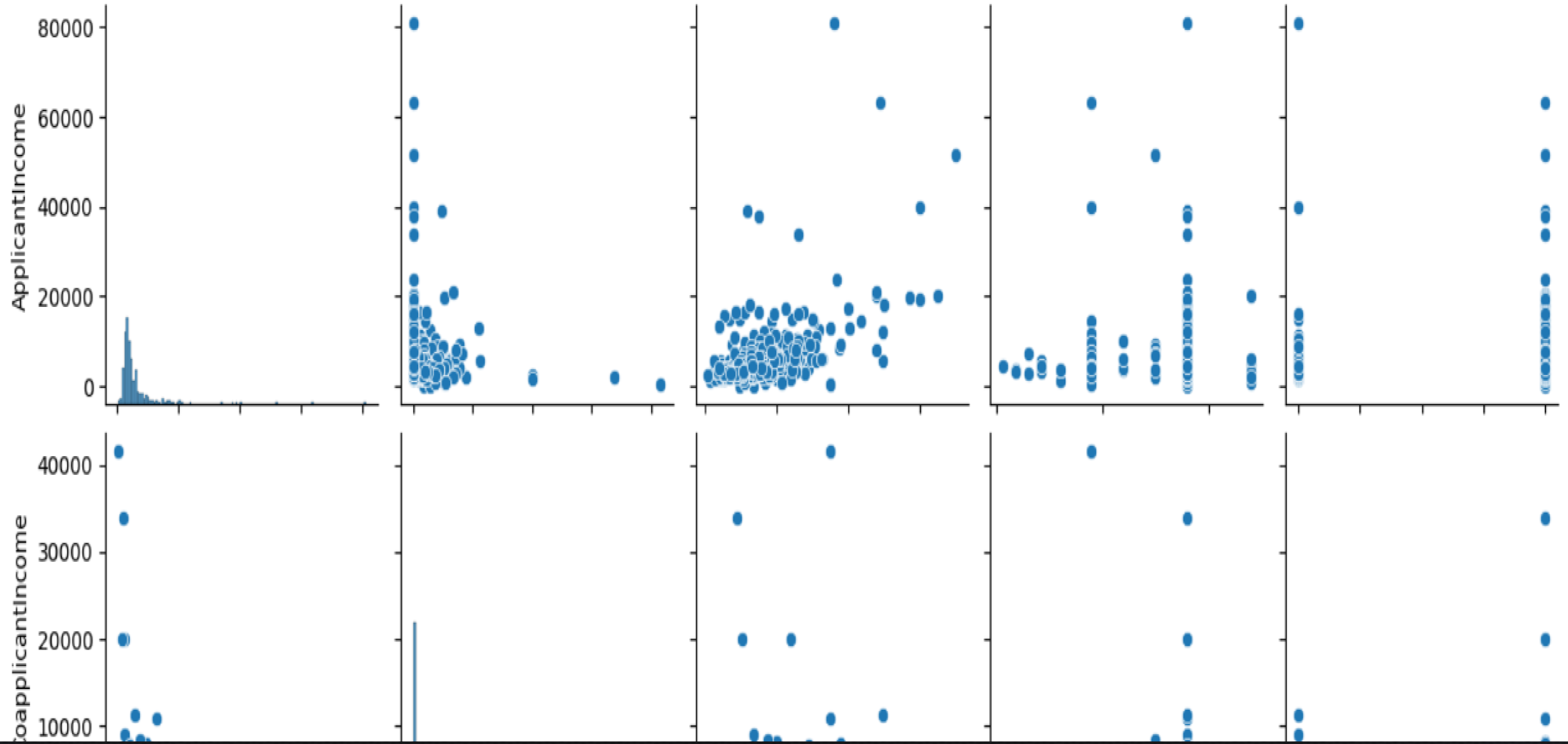
TASK 2: DEFINING EXPLORATORY DATA ANALYSIS WITH AN OVERVIEW OF THE WHOLE PROJECT

In [610]:
```
data.describe()
```

Out[610]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

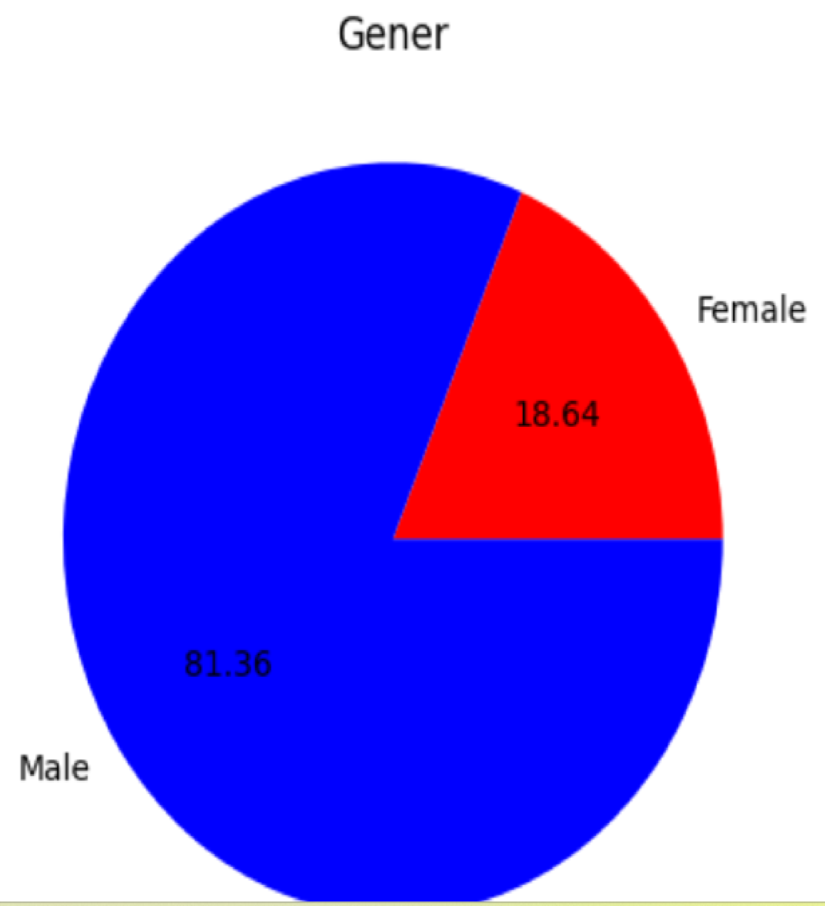
```
In [611]: sns.pairplot(data)

Out[611]: <seaborn.axisgrid.PairGrid at 0x28c30083400>
```



- **Data Visualization using piechart and countplot for categorical attributes (columns).**

In [612]:
```
data.groupby('Gender').size().plot(kind='pie', autopct='%.2f', colors=['red', 'blue'], title="Gener")
```

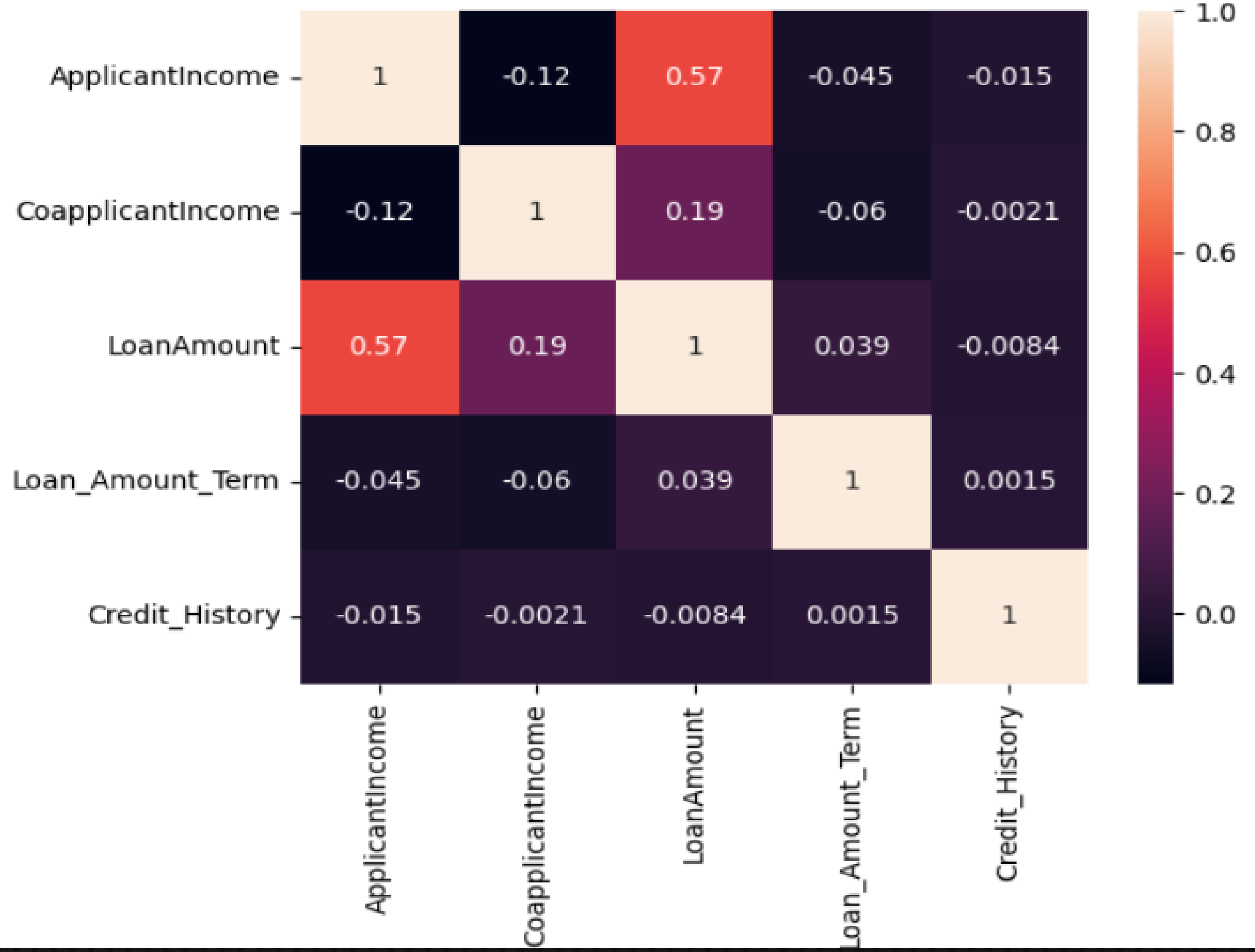
Out[612]:
```
<AxesSubplot: title={'center': 'Gener'}>
```



- **Matrix form for correlation data "Heatmap"**

```
In [618]: data.corr()
          sns.heatmap(data.corr(),annot=True)
```

```
Out[618]: <AxesSubplot: >
```



- # Task 3: Checking missing values and Outliers & Creating visual methods to analyze the data.

Check the nulls in data

In [620]:
```
data.isnull().sum()
```

Out[620]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Handle categorical missing data

- We replace missing data with mode

TASK 4: CREATE A MODEL THAT FITS THE DATA

Convert the categorical data into numerical data

- Now, we must encode the data which means converting the categorical variables into a numeric form to convert it to a machine-readable form, and this can be done through using LabelEncoder () from Sklearn.preprocessing library, and also using OrdinalEncoder() from Sklearn.preprocessing library also.
- There are a lot of ways to convert the data into numerical data but I will mention these two ways Only.

In [631]:
```
# Label Encode The Target Variable
encode = LabelEncoder()
data.Loan_Status = encode.fit_transform(data.Loan_Status)
```

In [632]:
```
# Ordinal Encode The features
enc = OrdinalEncoder()
data[["Gender",'Married','Education','Self_Employed','Property_Area','Loan_Status', 'Dependents']] = enc.fit_transform(data[["Ger
data.head()
```

Out[632]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	1.0	0.0	0.0	0.0	0.0	5849.0	0.0	128.0	360.0	1.0
1	LP001003	1.0	1.0	1.0	0.0	0.0	4583.0	1508.0	128.0	360.0	1.0
2	LP001005	1.0	1.0	0.0	0.0	1.0	3000.0	0.0	66.0	360.0	1.0
3	LP001006	1.0	1.0	0.0	1.0	0.0	2583.0	2358.0	120.0	360.0	1.0
4	LP001008	1.0	0.0	0.0	0.0	0.0	6000.0	0.0	141.0	360.0	1.0

LOGISTIC REGRESSION MODEL

In [638]:
```
LR = LogisticRegression()
LR.fit(x_train,y_train)
predict = LR.predict(x_test)
print(classification_report(y_test, predict))
LRAcc = accuracy_score(predict,y_test)
print('Logistic Regression accuracy is: {:.2f}%'.format(LRAcc*100))
```

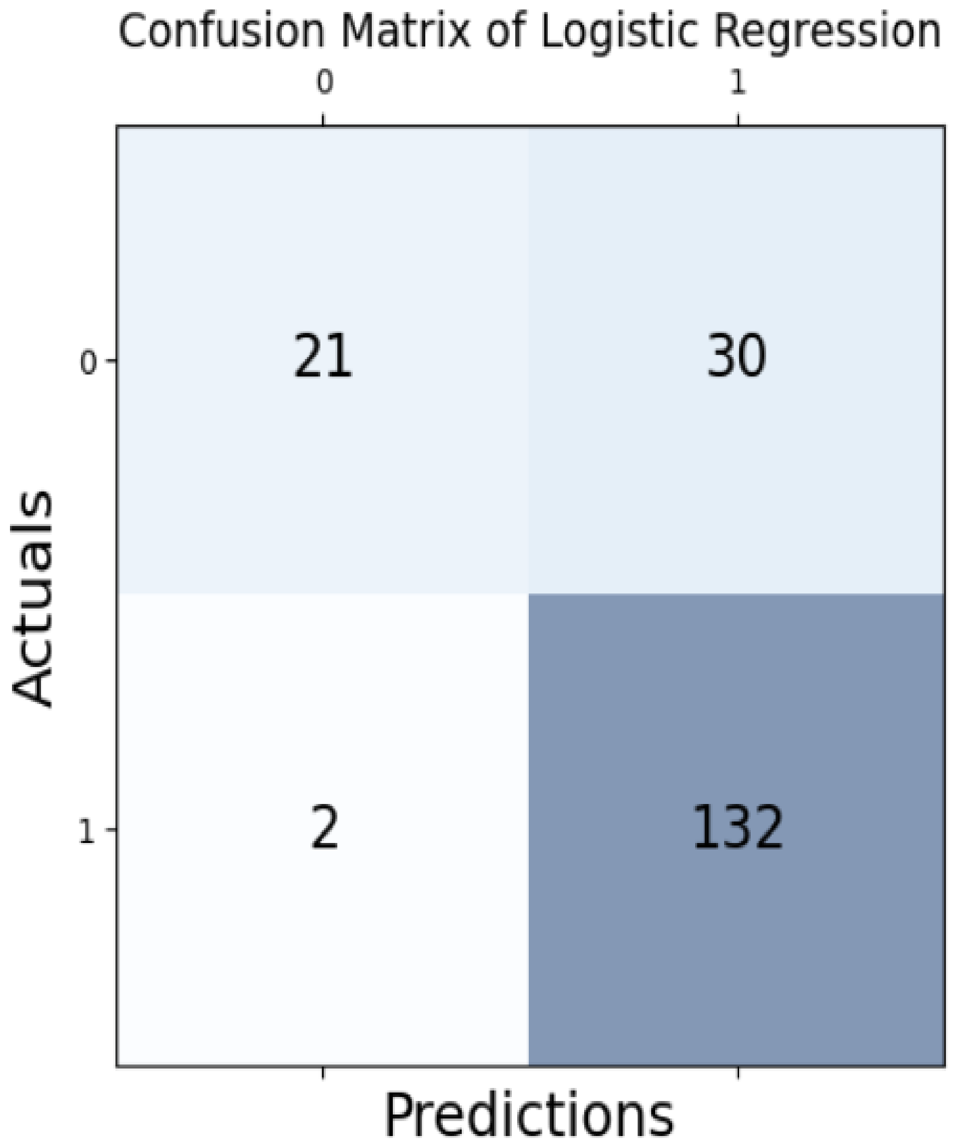
	precision	recall	f1-score	support
0.0	0.91	0.41	0.57	51
1.0	0.81	0.99	0.89	134
accuracy			0.83	185
macro avg	0.86	0.70	0.73	185
weighted avg	0.84	0.83	0.80	185

```
Logistic Regression accuracy is: 82.70%
```

CONFUSION MATRIX

```
In [639]: # Confusion Matrix for Logistic Regression
cm = metrics.confusion_matrix(y_test, predict)
print('Confusion Matrix for Logistic Regression :\n', cm, '\n')
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.5)
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix of Logistic Regression', fontsize=14)
plt.show()
```

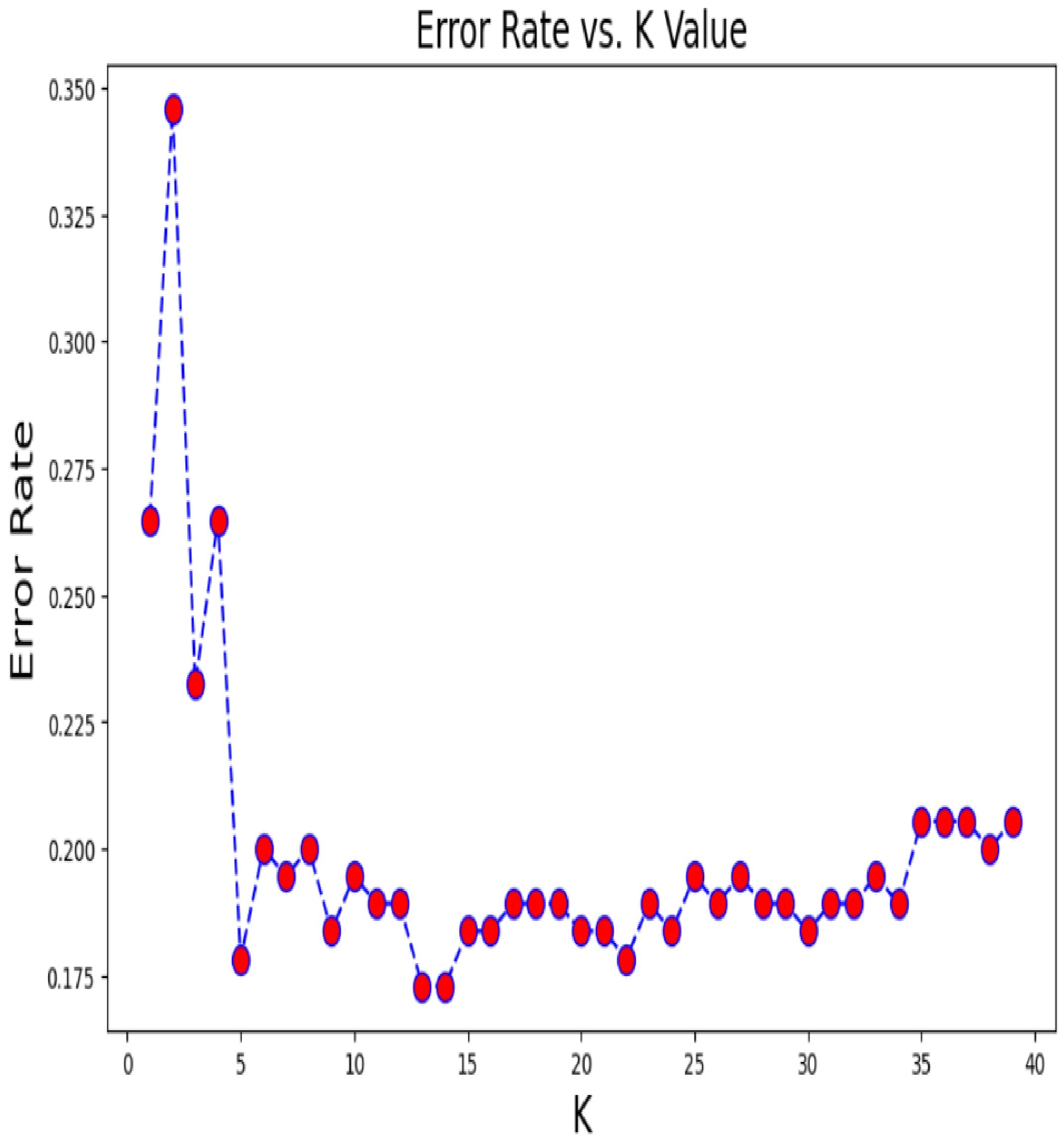
```
Confusion Matrix for Logistic Regression :
 [[ 21  30]
 [  2 132]]
```



KNN MODEL

To find a optimum value of K we plot a graph of error rate vs K value ranging from 0 to 40

In [640]:
```
error_rate = []
for i in range(1,40):
    kNN = KNeighborsClassifier(n_neighbors=i)
    kNN.fit(x_train,y_train)
    predict_i = kNN.predict(x_test)
    error_rate.append(np.mean(predict_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value', fontsize=18)
plt.xlabel('K', fontsize=18)
plt.ylabel('Error Rate', fontsize=18)
plt.show()
```

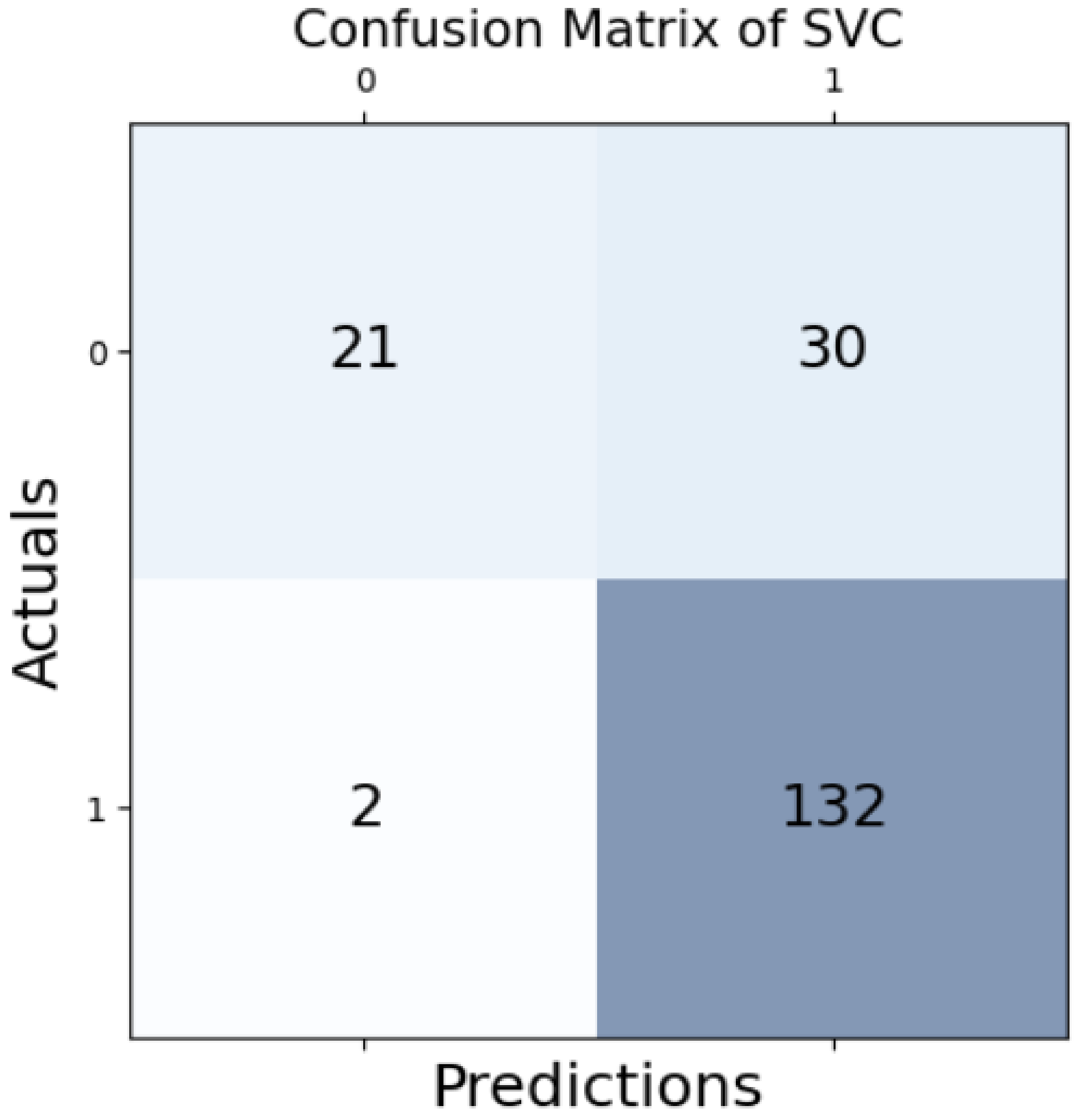


SVC MODEL

Confusion Matrix for SVC Model

In [644]:
```
cm = metrics.confusion_matrix(y_test, predict_svc)
print('Confusion Matrix for SVC :\n', cm, '\n')
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.5)
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix of SVC', fontsize=15)
plt.show()
```

```
Confusion Matrix for SVC :
 [[ 21  30]
 [  2 132]]
```



RANDOM FOREST MODEL

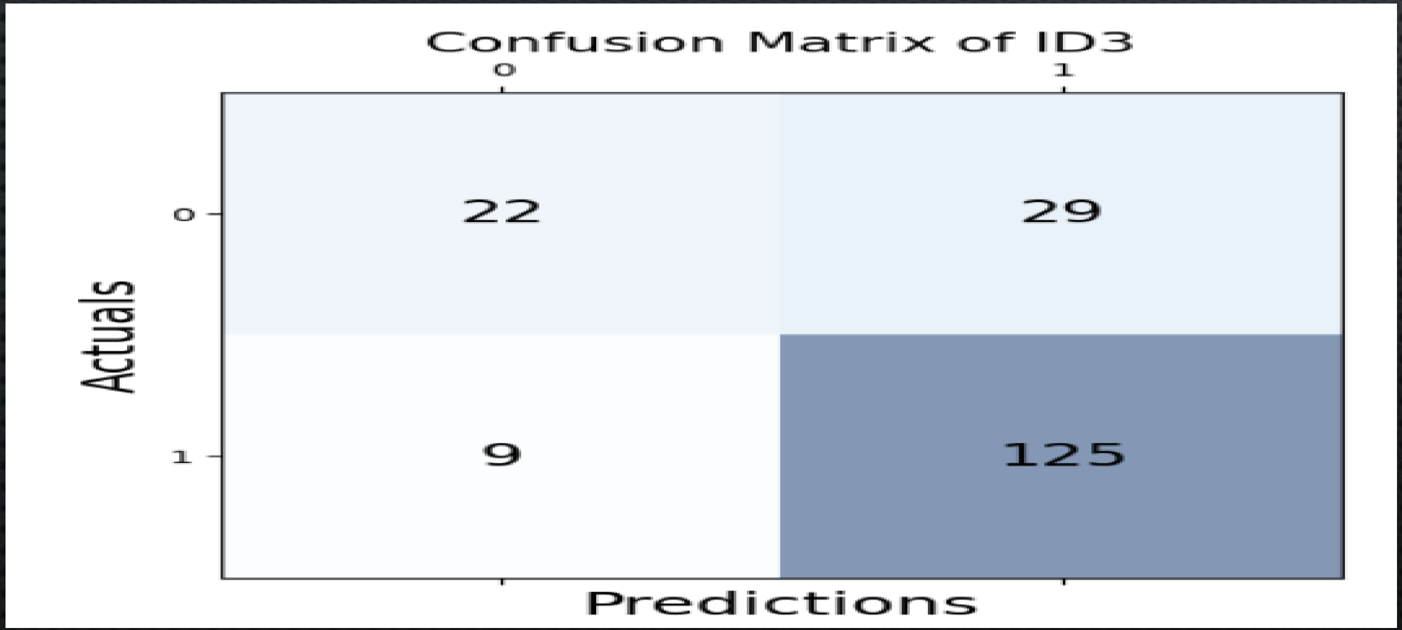
```
In [647]: clf=RandomForestClassifier(n_estimators=800)
          clf.fit(x_train,y_train)
          y_pred_rf=clf.predict(x_test)
          print(classification_report(y_test, y_pred_rf))
          rfAcc = accuracy_score(y_pred_rf,y_test)
          print('ID3 model accuracy is: {:.2f}%'.format(rfAcc*100))
```

```
              precision    recall  f1-score   support

         0.0       0.71      0.43      0.54        51
         1.0       0.81      0.93      0.87       134

    accuracy                           0.79       185
   macro avg       0.76      0.68      0.70       185
weighted avg       0.78      0.79      0.78       185

ID3 model accuracy is: 79.46%
```

Confusion Matrix for Random Forest

```
In [648]: cm = metrics.confusion_matrix(y_test, y_pred_rf)
          print('Confusion Matrix for ID3 :\n', cm, '\n')
          fig, ax = plt.subplots(figsize=(5, 5))
          ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.5)
          for i in range(cm.shape[0]):
              for j in range(cm.shape[1]):
                  ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='xx-large')

          plt.xlabel('Predictions', fontsize=18)
          plt.ylabel('Actuals', fontsize=18)
          plt.title('Confusion Matrix of ID3', fontsize=15)
          plt.show()
```

```
Confusion Matrix for ID3 :
 [[ 22  29]
 [  9 125]]
```



LOAD THE LOGISTIC REGRESSION MODEL

Load Random Forest Model with Joblib

In [649]:
```
joblib_file = "loan_predition_model_RF"
joblib.dump(clf, joblib_file)
loaded_model = joblib.load(open(joblib_file, 'rb'))
pred_y = loaded_model.predict(x_test)
result = np.round(accuracy_score(y_test, pred_y), 2)
print(result)
```

```
0.79
```

Load Logistic Regression model with Pickle

In [650]:
```
file = "loan_predition_model_LR.pkl"
pickle.dump(LR, open(file, 'wb'))

loaded_model = pickle.load(open(file, 'rb'))

pred_Y = loaded_model.predict(x_test)
result = np.round(accuracy_score(y_test, predict) ,2)
print(result)
```

```
0.83
```

• Conclusion :

From previos code we will notice we've chose Logistic Regression model to load and that because this model make the best prediction as the accuracy of it is the highest. but we are going to load Random Forest either to just comparing the two models (The highest and the lowest).

We now done. I hope this simple project will make you feel satisfaction with your hard work during this workshop and I hope I explain every point in this notebook and being a reference for my fellow students.