Data Cleaning and Preprocessing:

1. **After loading the data, we searched for duplicated data, dropped them, and replaced null values with zero.**
2. **we dropped 2 columns "Customer ID, Payment Method" as we won't use them.**
3. **To avoid any overflow we scaled our data using MinMaxScaler.**
4. **We changed the categories data to numerical.**
5. **We split the data into target and features then into train and test.**

Used Algorithms:

1. **Logistic Regression:**
2. **SVM**
3. **Decision Tree Classifier**

Accuracy of each algorithm:

	Models	Accuracy
0	LR	0.807312
1	SVM	0.803871
2	DT	0.720860

After algorithms and getting their accuracy we made a module for each one to predict using the data user enters.

GUI:

Using Tkinter library:

- **The First window allows the user to enter data to predict and has buttons:**
  1. **Predict: display a message box containing the prediction using all algorithms.**
  2. **Reset: clear all entry fields that the user entered data in them.**
  3. **Train: display a message box containing all data relevant to training data.**
  4. **Test: display a message box containing all data relevant to Testing data.**
- **Data Visualization button display a second window containing eight buttons:**
  1. **CustomersDataset[Churn]: display a graph using a count plot from 'seaborn library'.**
  2. **Confusion Matrix of Logistic Regression: display a graph using a heatmap from 'seaborn library'.**
  3. **Confusion Matrix of SVM: display a graph using heatmap from 'seaborn library'.**
  4. **Confusion Matrix of DecisionTree: display a graph using heatmap from 'seaborn library'.**
  5. **Final Data Accuracy: display graph using barplot from 'seaborn library'**
  6. **Final Data Precision: display graph using barplot from 'seaborn library'**
  7. **Final Data Recall: display graph using barplot from 'seaborn library'**

8. **Final Data F1_Score: display graph using barplot from 'seaborn library'**

- **The source code of the project**

```
CustomersDataset = pd.read_csv("CustomersDataset.csv")

CustomersDataset.drop_duplicates(inplace=True)

CustomersDataset["Churn"] = CustomersDataset["Churn"].replace(['No', 'Yes'], [0, 1])
CustomersDataset["TotalCharges"] = CustomersDataset["TotalCharges"].replace([' '], [0])
CustomersDataset["PaperlessBilling"] = CustomersDataset["PaperlessBilling"].replace(['No', 'Yes'], [0, 1])
CustomersDataset["StreamingMovies"] = CustomersDataset["StreamingMovies"].replace(['No', 'Yes', 'No internet service'], [0, 1, 2])

CustomersDataset.drop(['customerID', 'PaymentMethod'], axis=1, inplace=True)
```

- **First line we load the data from the 'CSV file'.**
- **Second line we drop all duplicated rows.**
- **Then We convert all categories data into numerical data for all columns.**

Last line we drop two columns we don't need "Customer ID, Payment Method".

```
scaler = MinMaxScaler()
CustomersDataset = pd.DataFrame(scaler.fit_transform(CustomersDataset), columns=CustomersDataset.columns)
x = CustomersDataset.drop(["Churn"], axis=1)
y = CustomersDataset["Churn"]
```

- **Here we scale the data and split it into features and targets.**

```
x_train, x_test, y_train, y_test = train_test_split(
 x, y, test_size=0.33, random_state=44, shuffle=True)
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test = np.array(x_test)
y_test = np.array(y_test)

y_train = y_train.reshape(y_train.shape[0], 1)
y_test = y_test.reshape(y_test.shape[0], 1)
```

- **Here we split the data into (train=0.67 and test=0.33) then convert them to an array using the NumPy library.**

```
LR_model = LogisticRegression(
    max_iter=1000, random_state=33, solver='saga', C=1.0)
LR_model.fit(x_train, y_train)
y_pred = LR_model.predict(x_test)
y_pred_prob = LR_model.predict_proba(x_test)
```

- **Here we started to use the algorithms starting with logistic regression.**
- **In the second line, we train the data using function fit().**
- **In the third line using function predict() we predict using x_test.**
- **In the last line using the function predict_proba() we store the probability of y_predict.**

```
svm = svm.SVC()
svm.fit(x_train, y_train)
y_pred_2 = svm.predict(x_test)
```

```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
y_pred_3 = dt.predict(x_test)
```

- **The first photo is for the SVM algorithm and the second one is for the decision tree algorithm and we did the same as logistic regression in them.**

```
final_data_Accuracy = pd.DataFrame({'Models': ['LR', 'SVM', 'DT'],
                                    'Accuracy': [accuracy_score(y_test, y_pred),
                                                 accuracy_score(y_test, y_pred_2), accuracy_score(y_test, y_pred_3)]})

final_data_Precision = pd.DataFrame({'Models': ['LR', 'SVM', 'ID3'],
                                     'PRECISION': [precision_score(y_test, y_pred),
                                                   precision_score(y_test, y_pred_2), precision_score(y_test, y_pred_3)]})

final_data_Recall = pd.DataFrame({'Models': ['LR', 'SVM', 'ID3'],
                                  'Recall': [recall_score(y_test, y_pred),
                                             recall_score(y_test, y_pred_2), recall_score(y_test, y_pred_3)]})

final_data_F1_Score = pd.DataFrame({'Models': ['LR', 'SVM', 'ID3'],
                                    'F1_Score': [f1_score(y_test, y_pred),
                                                 f1_score(y_test, y_pred_2), f1_score(y_test, y_pred_3)]})
```

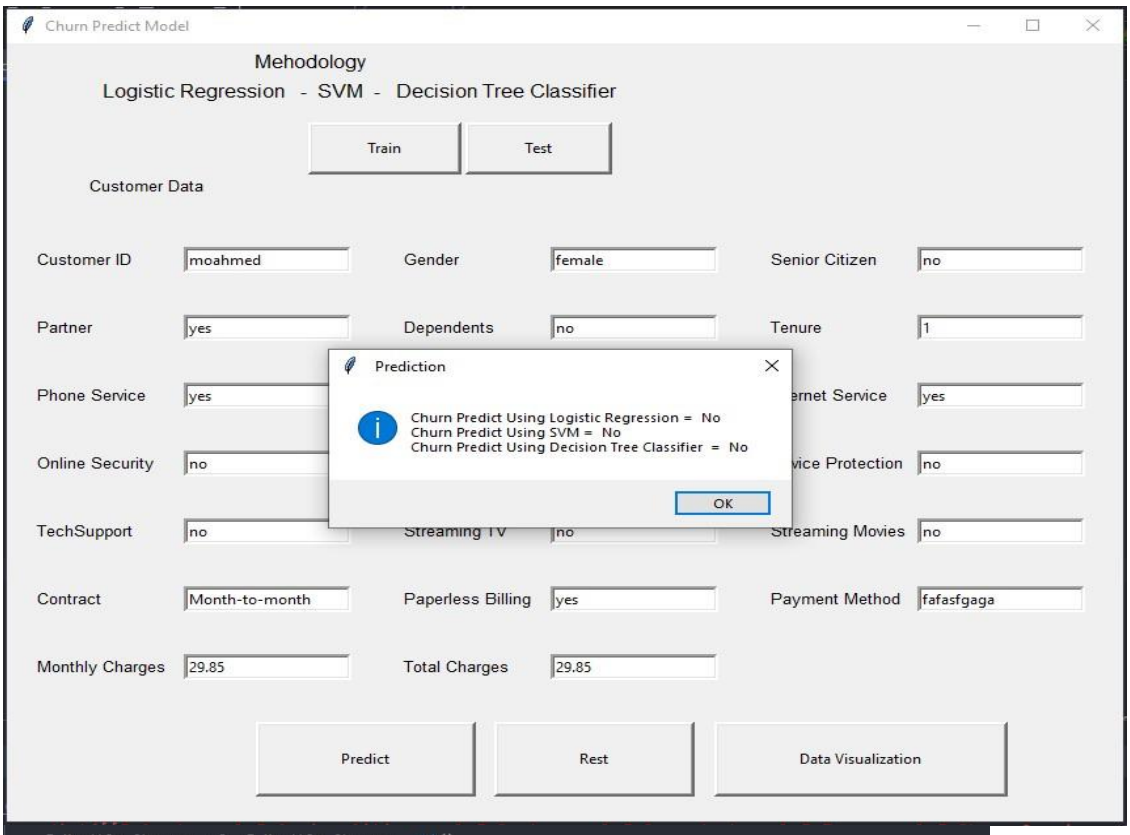
- **Here we collect "Accuracy Score, Precision Score, Recall Score, F1_Score" for all algorithms "LR, SVM, DT".**

```
lr_model = joblib.dump(LR_model, 'Churn Predict Model')
svm_model = joblib.dump(svm, 'Churn Predict Model')
dt_model = joblib.dump(dt, 'Churn Predict Model')

lr_model = joblib.load('Churn Predict Model')
svm_model = joblib.load('Churn Predict Model')
dt_model = joblib.load('Churn Predict Model')
```

- **Here we make a module for all algorithms to predict using them.**

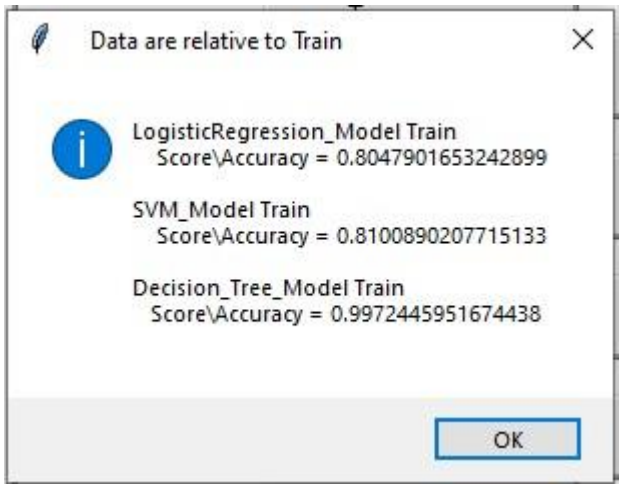
```
print('Predict Using Logistic Regression', lr_model.predict(
    [[0.0, 0.0, 1.0, 0.0, 1, 0.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 29.85, 29.85]]))
print('Predict Using SVM', svm_model.predict(
    [[0.0, 0.0, 1.0, 0.0, 1, 0.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 29.85, 29.85]]))
print('Predict Using Decision Tree Classifier', dt_model.predict(
    [[0.0, 0.0, 1.0, 0.0, 1, 0.0, 1.0, 0.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 29.85, 29.85]]))
```

- **This is an example of using the module to load input data to predict according to these data.**



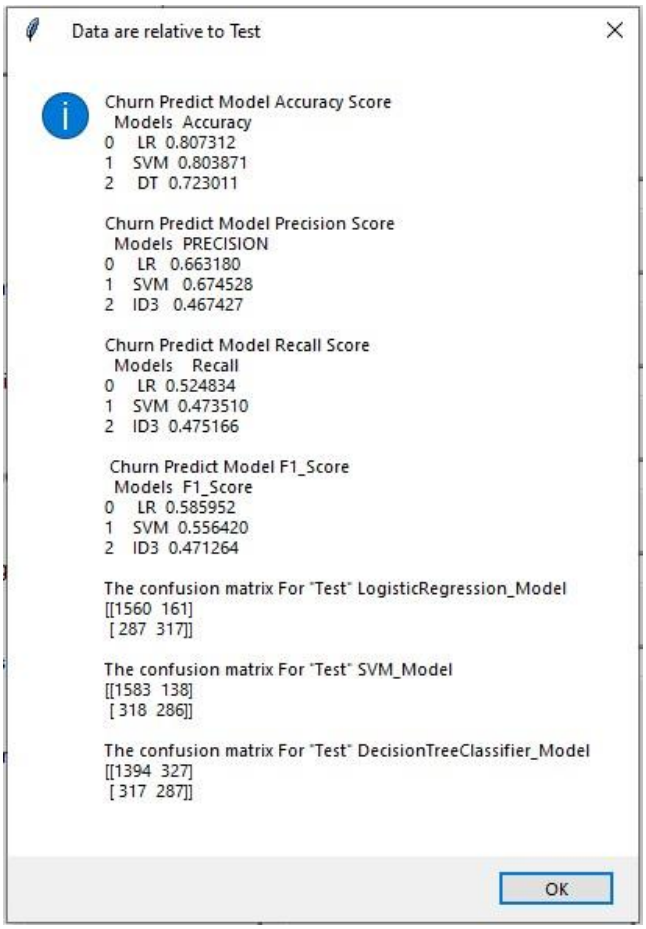

```
E_Monthly_Charges_val =E_Monthly_Charges .get()
E_Total_Charges_val =E_Total_Charges.get()
E_Tenure_val = E_Tenure.get()

# E_Partner
if E_Partner.get()=="no" or E_Partner.get()=="No" or E_Partner.get()=="NO":
    E_Partner_val =0
if E_Partner.get()=="yes" or E_Partner.get()=="Yes" or E_Partner.get()=="YES":
    E_Partner_val =1 ##############

# E_Dependents
if E_Dependents.get()=="no" or E_Dependents.get()=="No" or E_Dependents.get()=="NO":
    E_Dependents_val =0
if E_Dependents.get()=="yes" or E_Dependents.get()=="Yes" or E_Dependents.get()=="YES":
    E_Dependents_val =1 ##############

# E_Phone_Service
if E_Phone_Service.get()=="no" or E_Phone_Service.get()=="No" or E_Phone_Service.get()=="NO":
    E_Phone_Service_val =0
if E_Phone_Service.get()=="yes" or E_Phone_Service.get()=="Yes" or E_Phone_Service.get()=="YES":
    E_Phone_Service_val =1 ##############
```

- **This is the code for handling all entry fields in this window and this window appear when we press predict button.**



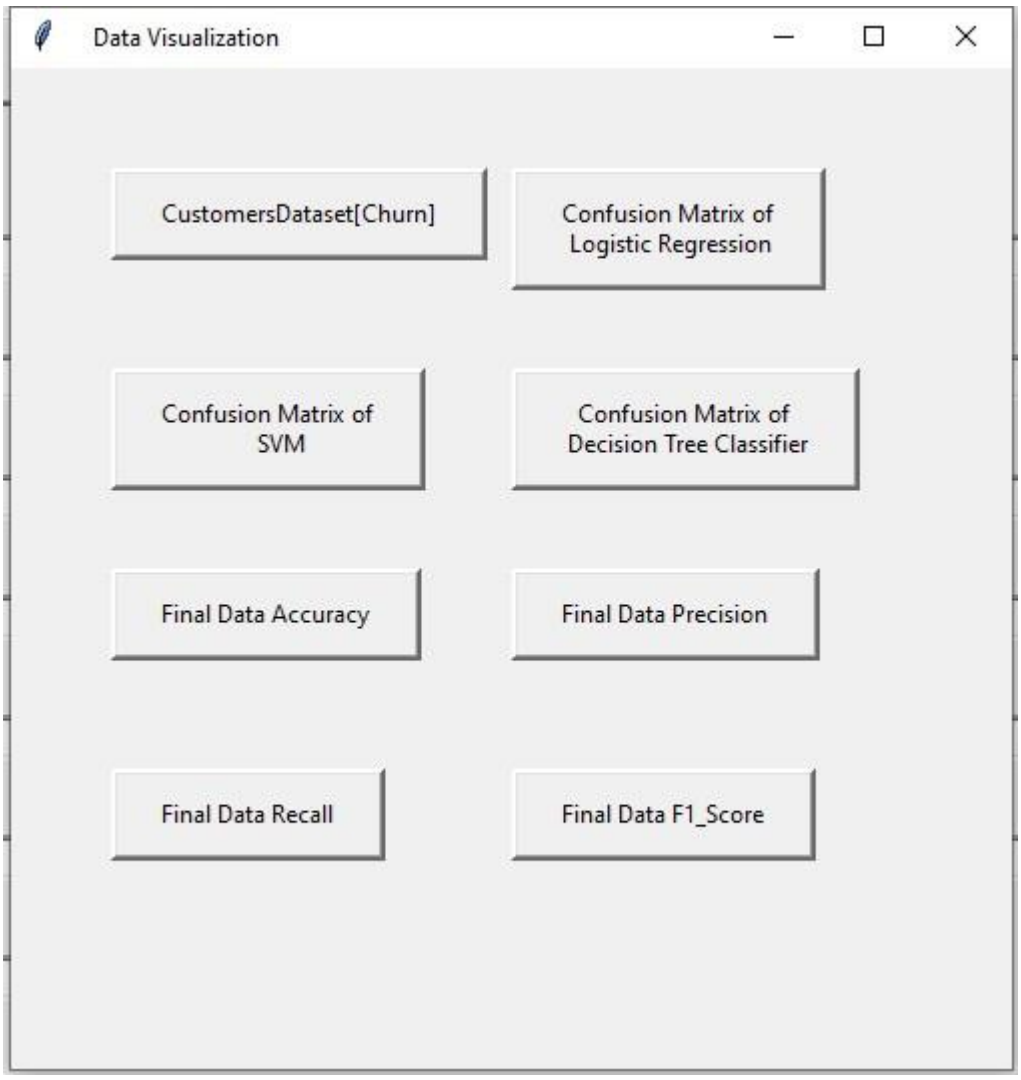
- **When we press the train button this window appears, And its code is below.**

```
def Display_Train():
    messagebox.showinfo("    Data are relative to Train",
    f"LogisticRegression_Model Train\n    Score\Accuracy = {LR_model.score(x_train,y_train)} \n\n"
    f"SVM_Model Train \n    Score\Accuracy = {svm.score(x_train,y_train)} \n\n"
    f"Decision_Tree_Model Train  \n   Score\Accuracy = {dt.score(x_train,y_train)}\n\n")
```



- **When we press the test button this window appears, And its code is below.**

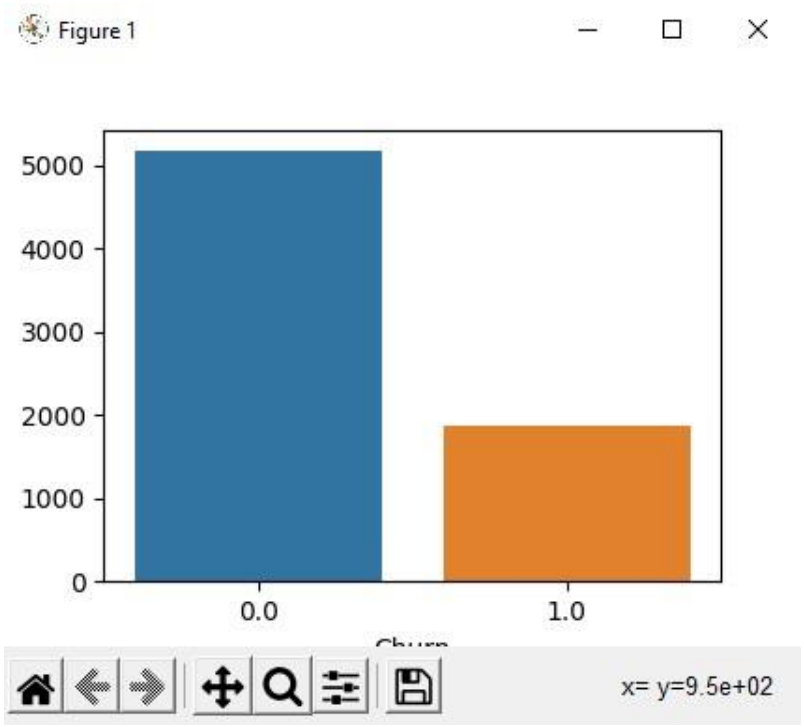
```
def Display_Test():
    messagebox.showinfo("    Data are relative to Test",f"Churn Predict Model Accuracy Score\n"
    f"{final_data_Accuracy}\n\nChurn Predict Model Precision Score\n{final_data_Precision}\n\n"
    f"Churn Predict Model Recall Score\n{final_data_Recall}\n\n Churn Predict Model F1_Score\n{final_data_F1_Score}\n\n"
    f"The confusion matrix For \"Test\" LogisticRegression_Model\n{confusion_matrix(y_test,y_pred)}\n\n"
    f"The confusion matrix For \"Test\" SVM_Model\n{confusion_matrix(y_test,y_pred_2)}\n\n"
    f"The confusion matrix For \"Test\" DecisionTreeClassifier_Model\n{confusion_matrix(y_test,y_pred_3)}\n\n")
```



- **This window appears when we press the data visualization button and its code below.**

```python
def Visualization():

    top = Toplevel()
    top.geometry("500x500")
    top.title("    Data Visualization  ")

    def dis_1():
        sns.countplot(CustomersDataset["Churn"])
        plt.show()

    def dis_2():
        sns.heatmap(confusion_matrix(y_test, y_pred), center=True)
        plt.show()

    def dis_3():
        sns.heatmap(confusion_matrix(y_test, y_pred_2), center=True)
        plt.show()

    def dis_4():
        sns.heatmap(confusion_matrix(y_test, y_pred_3), center=True)
        plt.show()
```

- **This barplot appears when we press CustomersDataset[Churn] button.**



```
Button(top,text="CustomersDataset[Churn]",padx=20,pady=10,borderwidth=2.5,command=dis_1).place(x=50,y=50)
Button(top,text="Confusion Matrix of\n Logistic Regression",padx=20,pady=10,borderwidth=2.5,command=dis_2).place(x=250,y=50)
Button(top,text="Confusion Matrix of\n     SVM",padx=20,pady=10,borderwidth=2.5,command=dis_3).place(x=50,y=150)
Button(top,text="Confusion Matrix of\n Decision Tree Classifier",padx=20,pady=10,borderwidth=2.5,command=dis_4).place(x=250,y=150)
Button(top,text="Final Data Accuracy",padx=20,pady=10,borderwidth=2.5,command=dis_5).place(x=50,y=250)
Button(top,text="Final Data Precision",padx=20,pady=10,borderwidth=2.5,command=dis_6).place(x=250,y=250)
Button(top,text="Final Data Recall",padx=20,pady=10,borderwidth=2.5,command=dis_7).place(x=50,y=350)
Button(top,text="Final Data F1_Score",padx=20,pady=10,borderwidth=2.5,command=dis_8).place(x=250,y=350)
```

- **This code for all data visualization and each line for one button from eight buttons in the data visualization window.**