Importing Dependencies

```
import pandas as pd
```

```
data = pd.read_csv('/content/train_u6lujuX_CVtuZ9i (1).csv')
```

```
# Loan_ID : Unique Loan ID

# Gender : Male/ Female

# Married : Applicant married (Y/N)

# Dependents : Number of dependents

# Education : Applicant Education (Graduate/ Under Graduate)

# Self_Employed : Self employed (Y/N)

# ApplicantIncome : Applicant income

# CoapplicantIncome : Coapplicant income

# LoanAmount : Loan amount in thousands of dollars

# Loan_Amount_Term : Term of loan in months

# Credit_History : Credit history meets guidelines yes or no

# Property_Area : Urban/ Semi Urban/ Rural

# Loan_Status : Loan approved (Y/N) this is the target variable
```

1. Displaying Top 5 rows in the Dataset

```
data.head()
```

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

2.Displaying last 5 rows in the Dataset

```
data.tail()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cr
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	

3. Find Shape of Our Dataset (Number of Rows And Number of Columns)

```
data.shape
```

```
(614, 13)
```

```
print('No of Rows :', data.shape[0])
print('No of Columns :', data.shape[1])
```

```
No of Rows : 614
No of Columns : 13
```

4. Get Information About Our Dataset Like Total Number Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

5. Check Null Values In The Dataset

```
data.isnull().sum()
```

```
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
```

```
# Getting %age of Null values in each columns
data.isnull().sum()*100/ len(data)
```

```
Loan_ID              0.000000
Gender               2.117264
Married              0.488599
Dependents           2.442997
Education            0.000000
Self_Employed        5.211726
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           3.583062
Loan_Amount_Term     2.280130
Credit_History       8.143322
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

6. Handling The missing Values

```
# Droping Loan_ID as it is not relevent
data =  data.drop('Loan_ID',axis=1)
```

```
data.head(1)
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0

```
columns = ['Gender','Dependents','LoanAmount','Loan_Amount_Term']
```

```
data = data.dropna(subset=columns)
```

```
data.isnull().sum()*100/ len(data)
```

```
Gender               0.000000
Married              0.000000
Dependents           0.000000
Education            0.000000
Self_Employed        5.424955
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.000000
Loan_Amount_Term     0.000000
```

```
Credit_History       8.679928
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

```
data['Self_Employed'].mode()[0]
```

```
'No'
```

```
data['Self_Employed'] =data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
```

```
data.isnull().sum()*100/len(data)
```

```
Gender               0.000000
Married              0.000000
Dependents           0.000000
Education            0.000000
Self_Employed        0.000000
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.000000
Loan_Amount_Term     0.000000
Credit_History       8.679928
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

```
data['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

```
data['Self_Employed'].unique()
```

```
array(['No', 'Yes'], dtype=object)
```

```
data['Credit_History'].unique()
```

```
array([ 1.,  0., nan])
```

```
data['Credit_History'].mode()[0]
```

```
1.0
```

```
data['Credit_History'] =data['Credit_History'].fillna(data['Credit_History'].mode()[0])
```

```
data.isnull().sum()*100 / len(data)
```

```
Gender               0.0
Married              0.0
Dependents           0.0
Education            0.0
Self_Employed        0.0
ApplicantIncome      0.0
CoapplicantIncome    0.0
LoanAmount           0.0
Loan_Amount_Term     0.0
Credit_History       0.0
Property_Area        0.0
Loan_Status          0.0
dtype: float64
```

7. Handling Categorical Columns

```
data.sample(5)
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
191	Male	No	0	Graduate	No	12000	0.0	164.0	360.0	
440	Male	No	0	Graduate	No	3660	5064.0	187.0	360.0	
542	Female	No	1	Graduate	No	3652	0.0	95.0	360.0	
241	Male	Yes	1	Not Graduate	No	2510	1983.0	140.0	180.0	
582	Female	Yes	0	Graduate	No	3166	0.0	36.0	360.0	

```
data['Dependents'] =data['Dependents'].replace(to_replace="3+",value='4')
```

```
data['Dependents'].unique()
```

```
array(['1', '0', '2', '4'], dtype=object)
```

```
data['Loan_Status'].unique()
```

```
array(['N', 'Y'], dtype=object)
```

```
data['Gender'] = data['Gender'].map({'Male':1,'Female':0}).astype('int')
data['Married'] = data['Married'].map({'Yes':1,'No':0}).astype('int')
data['Education'] = data['Education'].map({'Graduate':1,'Not Graduate':0}).astype('int')
data['Self_Employed'] = data['Self_Employed'].map({'Yes':1,'No':0}).astype('int')
data['Property_Area'] = data['Property_Area'].map({'Rural':0,'Semiurban':2,'Urban':1}).astype('int')
data['Loan_Status'] = data['Loan_Status'].map({'Y':1,'N':0}).astype('int')
```

```
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
1	1	1	1	1	0	4583	1508.0	128.0	360.0	1.0
2	1	1	0	1	1	3000	0.0	66.0	360.0	1.0
3	1	1	0	0	0	2583	2358.0	120.0	360.0	1.0
4	1	0	0	1	0	6000	0.0	141.0	360.0	1.0
5	1	1	2	1	1	5417	4196.0	267.0	360.0	1.0

8. Store Feature Matrix In X And Response (Target) In Vector y

```
X = data.drop('Loan_Status',axis=1)
```

```
y = data['Loan_Status']
```

```
y
```

```
1      0
2      1
3      1
4      1
5      1
      ..
```

```
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 553, dtype: int64
```

9. Feature Scaling

```
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
1	1	1	1	1	0	4583	1508.0	128.0	360.0	1.0
2	1	1	0	1	1	3000	0.0	66.0	360.0	1.0
3	1	1	0	0	0	2583	2358.0	120.0	360.0	1.0
4	1	0	0	1	0	6000	0.0	141.0	360.0	1.0
5	1	1	2	1	1	5417	4196.0	267.0	360.0	1.0

```
cols = ['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']
```

```
from sklearn.preprocessing import StandardScaler
st = StandardScaler()
X[cols]=st.fit_transform(X[cols])
```

```
X
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
1	1	1	1	1	0	-0.128694	-0.049699	-0.214368	0.279961	
2	1	1	0	1	1	-0.394296	-0.545638	-0.952675	0.279961	
3	1	1	0	0	0	-0.464262	0.229842	-0.309634	0.279961	
4	1	0	0	1	0	0.109057	-0.545638	-0.059562	0.279961	
5	1	1	2	1	1	0.011239	0.834309	1.440866	0.279961	
613	0	0	0	1	1	-0.128694	-0.545638	-0.154828	0.279961	

10. Splitting The Dataset Into The Training Set And Test Set & Applying K-Fold Cross Validation

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import numpy as np
```

```
model_df={}
def model_val(model,X,y):
    X_train,X_test,y_train,y_test=train_test_split(X,y,
                                                   test_size=0.20,
                                                   random_state=42)
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    print(f"{model} accuracy is {accuracy_score(y_test,y_pred)}")

    score = cross_val_score(model,X,y,cv=5)
    print(f"{model} Avg cross val score is {np.mean(score)}")
    model_df[model]=round(np.mean(score)*100,2)
```

```
model_df
```

    {}

11. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model_val(model,X,y)
```

```
LogisticRegression() accuracy is 0.8018018018018018
LogisticRegression() Avg cross val score is 0.8047829647829647
```

12. SVC

```
from sklearn import svm
model = svm.SVC()
model_val(model,X,y)
```

```
SVC() accuracy is 0.7927927927927928
SVC() Avg cross val score is 0.7938902538902539
```

13. Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model_val(model,X,y)
```

```
DecisionTreeClassifier() accuracy is 0.7477477477477478
DecisionTreeClassifier() Avg cross val score is 0.7106797706797707
```

14. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
model =RandomForestClassifier()
model_val(model,X,y)
```

```
RandomForestClassifier() accuracy is 0.7747747747747747
RandomForestClassifier() Avg cross val score is 0.7848484848484848
```

15. Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier
model =GradientBoostingClassifier()
model_val(model,X,y)
```

```
GradientBoostingClassifier() accuracy is 0.7927927927927928
GradientBoostingClassifier() Avg cross val score is 0.7721539721539721
```

16. Hyperparameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV
```

Logistic Regression

```
log_reg_grid= {"C": np.logspace(-4,4,20),
               "solver":['liblinear']}
```

```
rs_log_reg=RandomizedSearchCV(LogisticRegression(),
                   param_distributions=log_reg_grid,
                  n_iter=20,cv=5,verbose=True)
```

```
rs_log_reg.fit(X,y)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
▸      RandomizedSearchCV
▸ estimator: LogisticRegression
     ▸ LogisticRegression
```

```
rs_log_reg.best_score_
```

```
0.8047829647829647
```

```
rs_log_reg.best_params_
```

```
{'solver': 'liblinear', 'C': 0.23357214690901212}
```

SVC

```
svc_grid = {'C':[0.25,0.50,0.75,1],"kernel":["linear"]}
```

```
rs_svc=RandomizedSearchCV(svm.SVC(),
                  param_distributions=svc_grid,
                   cv=5,
```

```
                   n_iter=20,
                  verbose=True)
```

```
rs_svc.fit(X,y)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_search.py:305: UserWarning: The total space of parameters 4 is smaller th
  warnings.warn(
Fitting 5 folds for each of 4 candidates, totalling 20 fits
```

RandomizedSearchCV

estimator: SVC

SVC

```
rs_svc.best_score_
```

```
0.8066011466011467
```

```
rs_svc.best_params_
```

```
{'kernel': 'linear', 'C': 0.25}
```

Random Forest Classifier

```
RandomForestClassifier()
```

RandomForestClassifier

RandomForestClassifier()

```
rf_grid={'n_estimators':np.arange(10,1000,10),
  'max_features':['auto','sqrt'],
 'max_depth':[None,3,5,10,20,30],
 'min_samples_split':[2,5,20,50,100],
 'min_samples_leaf':[1,2,5,10]
 }
```

```
rs_rf=RandomizedSearchCV(RandomForestClassifier(),
                  param_distributions=rf_grid,
                   cv=5,
```

```
                  n_iter=20,
                 verbose=True)


rs_rf.fit(X,y)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
  warn(
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
```

```
warn(
```

```
rs_rf.best_score_
```

```
0.8066175266175266
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
```

```
rs_rf.best_params_
```

```
{'n_estimators': 760,
 'min_samples_split': 20,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 5}
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
```

LogisticRegression score Before Hyperparameter Tuning: 80.48

LogisticRegression score after Hyperparameter Tuning: 80.48

SVC score Before Hyperparameter Tuning: 79.38

SVC score after Hyperparameter Tuning: 80.66

RandomForestClassifier score Before Hyperparameter Tuning: 77.76

RandomForestClassifier score after Hyperparameter Tuning: 80.66

```
/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_fe
```

Save Model

```
X = data.drop('Loan_Status',axis=1)
y = data['Loan_Status']
```

```
rf = RandomForestClassifier(n_estimators=270,
 min_samples_split=5,
 min_samples_leaf=5,
 max_features='sqrt',
 max_depth=5)
```

```
RandomForestClassifier
```

```
rf.fit(X,y)
```

RandomForestClassifier

```
import joblib
```

```
joblib.dump(rf,'loan_status_predict')
```

    ['loan_status_predict']

```
model = joblib.load('loan_status_predict')
```

```
import pandas as pd
df = pd.DataFrame({
    'Gender':1,
    'Married':1,
    'Dependents':2,
    'Education':0,
    'Self_Employed':0,
    'ApplicantIncome':2889,
    'CoapplicantIncome':0.0,
    'LoanAmount':45,
    'Loan_Amount_Term':180,
    'Credit_History':0,
    'Property_Area':1
},index=[0])
```

```
df
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	1	1	2	0	0	2889	0.0	45	180	0

```
result = model.predict(df)
```

```
if result==1:
    print("Loan Approved")
else:
    print("Loan Not Approved")
```

    Loan Not Approved

GUI

```
from tkinter import *
import joblib
import pandas as pd


def show_entry():

    p1 = float(e1.get())
    p2 = float(e2.get())
    p3 = float(e3.get())
    p4 = float(e4.get())
    p5 = float(e5.get())
    p6 = float(e6.get())
    p7 = float(e7.get())
    p8 = float(e8.get())
    p9 = float(e9.get())
    p10 = float(e10.get())
    p11 = float(e11.get())

    model = joblib.load('loan_status_predict')
    df = pd.DataFrame({
    'Gender':p1,
    'Married':p2,
    'Dependents':p3,
    'Education':p4,
    'Self_Employed':p5,
    'ApplicantIncome':p6,
    'CoapplicantIncome':p7,
    'LoanAmount':p8,
    'Loan_Amount_Term':p9,
    'Credit_History':p10,
    'Property_Area':p11
},index=[0])
    result = model.predict(df)

    if result == 1:
        Label(master, text="Loan approved").grid(row=31)
    else:
        Label(master, text="Loan Not Approved").grid(row=31)



master =Tk()
master.title("Loan Status Prediction Using Machine Learning")
label = Label(master,text = "Loan Status Prediction",bg = "black",
               fg = "white").grid(row=0,columnspan=2)
```

```
Label(master,text = "Gender [1:Male ,0:Female]").grid(row=1)
Label(master,text = "Married [1:Yes,0:No]").grid(row=2)
Label(master,text = "Dependents [1,2,3,4]").grid(row=3)
Label(master,text = "Education").grid(row=4)
Label(master,text = "Self_Employed").grid(row=5)
Label(master,text = "ApplicantIncome").grid(row=6)
Label(master,text = "CoapplicantIncome").grid(row=7)
Label(master,text = "LoanAmount").grid(row=8)
Label(master,text = "Loan_Amount_Term").grid(row=9)
Label(master,text = "Credit_History").grid(row=10)
Label(master,text = "Property_Area").grid(row=11)


e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
e8 = Entry(master)
e9 = Entry(master)
e10 = Entry(master)
e11 = Entry(master)


e1.grid(row=1,column=1)
e2.grid(row=2,column=1)
e3.grid(row=3,column=1)
e4.grid(row=4,column=1)
e5.grid(row=5,column=1)
e6.grid(row=6,column=1)
e7.grid(row=7,column=1)
e8.grid(row=8,column=1)
e9.grid(row=9,column=1)
e10.grid(row=10,column=1)
e11.grid(row=11,column=1)

Button(master,text="Predict",command=show_entry).grid()

mainloop()
```

```
---------------------------------------------------------------------------
TclError                                  Traceback (most recent call last)
<ipython-input-75-5a835c3cf813> in <module>
     35 
     36 
---> 37 master =Tk()
     38 master.title("Loan Status Prediction Using Machine Learning")
     39 label = Label(master,text = "Loan Status Prediction",bg = "black",

/usr/lib/python3.8/tkinter/__init__.py in __init__(self, screenName, baseName, className, useTk, sync, use)
   2268                 baseName = baseName + ext
   2269         interactive = 0
-> 2270         self.tk = _tkinter.create(screenName, baseName, className, interactive, wantobjects, useTk, sync, use)
   2271         if useTk:
   2272             self._loadtk()

TclError: no display name and no $DISPLAY environment variable
```

SEARCH STACK OVERFLOW