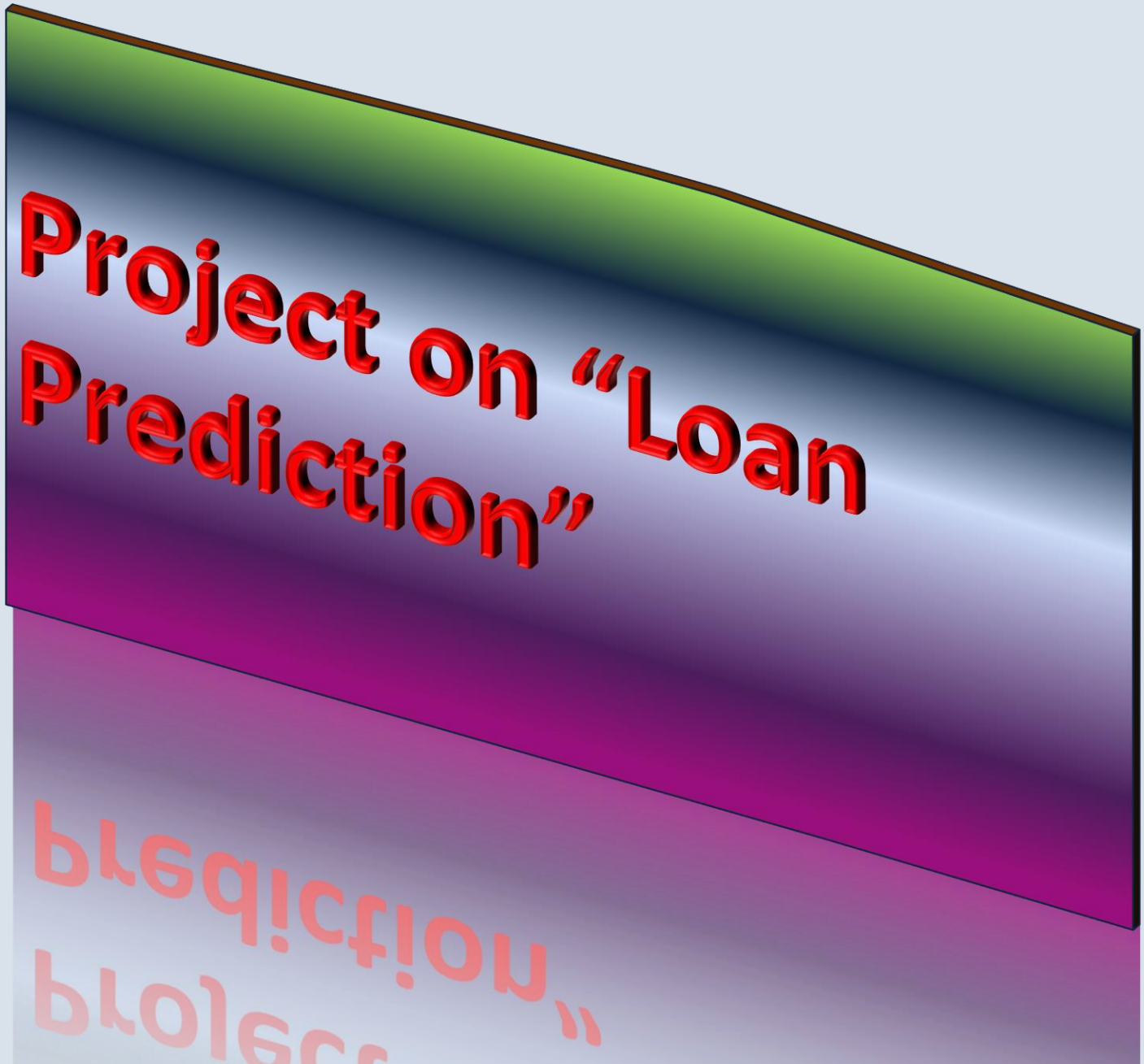
Data Science with Python Programming

- Presentation By Uplatz
- Contact us: https://training.uplatz.com
- Email: info@uplatz.com
- Phone: +44 7836 212635

Uplatz



Uplatz

Learning outcomes:

- Problem Statement
- Dataset
- Exploratory Data Analysis
- Implementation of Project

Problem Statement

Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

The company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others.

Problem Statement

To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers.
So the final thing is to identify the factors/ customer segments that are eligible for taking loan.
How will the company benefit if we give the customer segments is the immediate question that arises. The solution is ….Banks would give loans to only those customers that are eligible so that they can be assured of getting the money back.

Problem Statement

Hence the more accurate we are in predicting the eligible customers the more beneficial it would be for the Dream Housing Finance Company.

Problem:

Predict if a loan will get approved or not.

This is a classification problem as we need to classify whether the **Loan_Status** is **yes** or **no**.

Dataset

Look at the dataset that we have for solving this problem.

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)

Dataset

Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	Loan approved (Y/N)

*There are altogether 13 columns in our data set. Of them **Loan_Status** is the response variable and rest all are the variables /factors that decide the approval of the loan or not.*

Uplatz

Exploratory Data Analysis

In statistics, **exploratory data analysis** is an approach to analysing data sets to summarize their main characteristics, often with visual methods. Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in-hand, before getting them dirty with it.

Exploratory Data Analysis

By looking at the columns description in dataset, we can make many assumptions like

- The one whose salary is more can have a greater chance of loan approval.
- The one who is graduate has a better chance of loan approval.
- Married people would have a upper hand than unmarried people for loan approval .
- The applicant who has less number of dependents have a high probability for loan approval.
- The lesser the loan amount the higher the chance for getting loan.

Implementation of Project

Now let’s walk through the code. Firstly I just imported the necessary packages like pandas, numpy, seaborn etc. so that i can carry the necessary operations further.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Implementation of Project

Read the data from the file/dataset:

```
df = pd.read_csv('F:\Alison R Studio\Loan.csv')
```

Data cleaning and filling messing values:

Let’s see the code for it.

Implementation of Project

Converting Categorical variable into numeric:

What is Label Encoding?

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Implementation of Project

Converting Categorical variable into numeric:

What is Label Encoding?

Suppose we have a column *Height* in some dataset.

Height
Tall
Medium
Short

After applying label encoding, the Height column is converted into:

Height
0
1
2

Uplatz

Implementation of Project

Converting Categorical variable into numeric:
sklearn requires all inputs to be numeric, we should convert all our categorical variables into numeric by encoding the categories.
And to convert this kind of categorical text data into model-understandable numerical data, we use the **LabelEncoder** class. So all we have to do, is to import the **LabelEncoder** class from the **sklearn** library, fit and transform the columns of the data.

Implementation of Project

Converting Categorical variable into numeric:
This can be done using the following code:

```
from sklearn.preprocessing import LabelEncoder
var_mod =
['Gender','Married','Dependents','Education','Self_
Employed','Property_Area','Loan_Status']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
```

Here **fit_transform()** Fit label encoder and return encoded labels.

Implementation of Project

Feature Selection:
Here, you need to divide given columns into two types of variables **dependent(or target variable)** and **independent variable(or feature variables)**.

```
X = df.iloc[:,1:12]
y = df.iloc[:,12]
```

The iloc indexer for Pandas Dataframe is used for integer-location based indexing / selection by position.

Implementation of Project

Splitting Data:

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. Let's split the dataset by using function **train_test_split().** You need to pass 4 parameters features, target, test_size, and random_sate.

```
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,random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)
```

Implementation of Project

Building Decision Tree Model:

Let's create a Decision Tree Model using Scikit-learn.

```
from sklearn.tree import DecisionTreeClassifier
# Create Decision Tree classifer object
model= DecisionTreeClassifier()
# Train Decision Tree Classifer
model.fit(X_train,y_train)
#Predict the response for test dataset
y_predictions= model.predict(X_test)
```

Implementation of Project

Evaluating Model:

Let's estimate, how accurately the classifier or model can predict the type of cultivars.

Accuracy can be computed by comparing actual test set values and predicted values.

Model Accuracy, how often is the classifier correct?

```
print("Accuracy:",metrics.accuracy_score(y_test, y_predictions))
```

Well, you got a classification rate of 68.29% using Decision tree algorithm.

Implementation of Project

Building Logistic Regression Model:

First, import the Logistic Regression module and create a Logistic Regression classifier object using **LogisticRegression()** function.

Then, fit your model on the train set using **fit()** and perform prediction on the test set using **predict().**

```
from sklearn.linear_model import
LogisticRegression
logistic_regression= LogisticRegression()
logistic_regression.fit(X_train,y_train)
y_pred=logistic_regression.predict(X_test)
```

Implementation of Project

Model Evaluation using Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(cnf_matrix)
```

Implementation of Project

Model Evaluation using Confusion Matrix:

Here, you can see the confusion matrix in the form of the array object. The dimension of this matrix is 2*2 because this model is binary classification. You have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, 15 and 88 are actual predictions, and 18 and 2 are incorrect predictions.

This means out of 123, we got 103 correct prediction and 20 incorrect prediction.

Implementation of Project

Visualizing Confusion Matrix using Heatmap

Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.

```
import seaborn as sn
sn.heatmap(cnf_matrix, annot=True)
plt.title('Confusion matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

Here **annot** – an array of same shape as data which is used to annotate the heatmap.

Implementation of Project

Model Evaluation:

You can also find the **Accuracy** by comparing actual test set values and predicted values.

Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Well, you got a classification rate of 83.73%(You might get some other value) using Logistic regression algorithm which is considered as good accuracy.

Implementation of Project

Let's predict whether the **loan will get approved or not for a person(John) who is applying for the loan with the following details:**

Gender : Male
Married: Yes
Dependents: 1
Education: Graduate
Self_Employed: No
ApplicantIncome: 8000
CoapplicantIncome: 2000
LoanAmount (in thousand): 130
Loan_Amount_Term(Term of loan in months): 24
Credit_History: 0.0
Property_Area (Urban/ Semi Urban/ Rural): Urban

Implementation of Project

Convert the categorical variable value into numeric form.

Gender : Male (1)
Married: Yes (1)
Dependents: 1
Education: Graduate (0)
Self_Employed: No (0)
ApplicantIncome: 8000
CoapplicantIncome: 2000
LoanAmount (in thousand): 130
Loan_Amount_Term(Term of loan in months): 24
Credit_History: 0.0
Property_Area (Urban/ Semi Urban/ Rural): Urban (2)

Implementation of Project

```
Loanstatus =
logistic_regression.predict([[1,1,1,0,0,8000,2000,1
30,24,0.0,2]])
print(loanstatus)
```

After the execution of this code, you can see the output as :

```
array([1])
```

This means that the loan get approved for the person (John) and he will get the loan.

Implementation of Project