Date – 21/09/2021

LOAN APPROVAL PREDICTION

Internship Project Report

Organized by: **Innovation. Creation (IC Solutions)**

Submitted by:

RANJITH N [1VA19CS041, ranjithn.19cs@saividya.ac.in]

MONICA P [1EW20IS048, monicaspgowda@gmail.com]

PUSHPA B M [1ME18CS048, pushpabm496@gmail.com]

VARSHITHA S [1HK19CS177, varshisuresh1742@gmail.com]

Under the guidance of: Mr. ABHISHEK C

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to Mr. **Abhishek C**, ML Tutor for providing me an opportunity to do my internship and project work in "**INNOVATION.CREATION**".

No words would suffice to express my regards and gratitude to the officials and other staff members of '**INNOVATION.CREATION**' for their inspiring guidance, and constant encouragement, immense support and help during the project work.

Last, but not the least, **my parents** are also an important inspiration for me. So with due regards, I express my gratitude to them.

Sincerely,
Ranjith N
Monica P
Pushpa B M
Varshitha S

Place: Bengaluru
Date: 21/09/2021

ABSTRACT

In today's world, taking loans from financial institutions has become a very common phenomenon. Everyday a large number of people make applications for loans, for a variety of purposes. But all these applications are not reliable and everyone cannot be approved. In our banking system, banks have many products to sell but main source of income of any banks is on its credit line. So, they can earn form interest of those loans which they credit. A bank's profit or a loss depends to a large extent on loans i.e., whether the customers are paying back the loan or defaulting. This makes the study of this phenomenon very important. This model can be used by organizations in making the right decision to approve or reject the loan request of the customers. The main objective of this project is to predict whether assigning the loan to particular person will be safe or not. This consists of several steps:

(i) Data Collection

(ii) Comparison of machine learning models on collected data

(iii) Training of system on most promising model

(iv) Testing.

In this project we are predicting the loan data by using some machine learning algorithms they are:

A. Linear Regression
B. Logistic Regression
C. Support Vector Machine
D. Random Forest
E. Decision Tree
F. Polynomial Regression

ABOUT THE COMPANY

IC SOLUTIONS(ICS) is a digital service provider that aims to provide Software designing and Marketing solutions to individuals and businesses.

At ICS, we believe Service and Quality are the key of success.

We provide all kinds of technological and designing from Billing software to Web designs or any custom demand that you may have.

Experience the service like none other!

Some of our services include:

- Development – We develop responsive, functional and super-fast Websites. We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small View port and slow Internet connection.
- Mobile Applications – We offer a wide range of professional Android, iOS & Hybrid development services for our global clients, from a startup to a large enterprise.
- Design – We offer professional Graphic, Brochure design & Logodesign. We are experts in creating visual content to convey the right message to the customers.
- Consultancy – We are here to provide you with expert advice on your Design and development requirement.
- Videos – We create a polished professional video that impress your Audience.

INDEX

INTRODUCTION

Loan Prediction is very helpful for employee of banks as well as for the applicant also. Loans are the core business of banks. The main portion of the banks assets directly come from the profit earned from the loans distributed by the banks. The loan companies grant a loan after an intensive process of verification and validation. The prime objective in banking environment is to invest their assets in safe hands where it is. Today many banks/financial companies approve loan after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of feature is automated by **Machine learning technique**.

The aim of this report is to provide quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. The **Loan prediction system** can automatically calculate the weight of each feature taking part in loan processing and on new test data. Whole processing of prediction is done privately so that no stakeholders would be able to alter the processing. Results against particular technique Loan Id can be sent to various departments of banks. This helps all other department to carry out other Formalities. We will prepare the data using Jupyter Notebook and use various models to predict the target variable.

ML PROJECT END-TO-END STEPS

1] DATA CLEANING AND FORMATTING:

Data cleansing or data cleaning is the process of identifying and removing inaccurate records from a dataset, table or database and refers to recognising unfinished, unreliable, inaccurate or non-relevant part of the data and then restoring, remodelling, or removing the dirty and crude data.

2] EXPLORATORY DATA ANALYSIS:

Exploratory data analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypotheses and check assumptions with the help of summary statistics and graphical representations.

It is an approach for summarizing, visualizing, and becoming intimately familiar with the important characteristics of a dataset.

3] FEATURE ENGINEERING AND SELECTION:

The well-known concept of "garbage in-garbage out" applies 100% to any task in ML.

4] FEATURE EXTRACTION AND FEATURE ENGINEERING:

Transformation of raw data into features suitable for modelling.

5] FEATURE TRANSFORMATIONS:

Transformation of data to improve the accuracy of the algorithm.

6] FEATURE SELECTION:

Removing unnecessary features.

ML PROJECT END-TO-END STEPS

7] COMPARE MULTIPLE ALGORITHMS:

It is important to compare the performance of multiple different machine learning algorithms consistently.

When you work on an ML project, you often end up with multiple good models to choose from, each model will have different performance characteristics.

In the example below 6 different algorithms are compared:

Linear Regression, Logistic Regression, SVM, Decision Tree, Random Forest, Polynomial Regression.

8] HYPERPARAMETER TUNING:

These are not model parameters and they can't be directly trained from the data.

In ML, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.

It is a parameter whose value is used to control the learning process.

9] EVALUATE THE MODELS:

Evaluating helps to find the best model which represents our data and how well the chosen model will work in the future.

Evaluating model performance with the data used for training is not acceptable in ML because it can easily generate over-optimistic and overfitted models.

There are two methods of evaluating models in ML hold out and cross-validation.

ML PROJECT END-TO-END STEPS

10] DEPLOY THE MODEL:

Model interpretability helps debug the model by what the model really thinks is important.

The purpose of deploying model is so that we can make the predictions from training ML model by making it available to the outside world.

11] CONCLUSIONS AND DOCUMENTATION:

This helps in reproducibility and also ensures successful project completion.

Documentation helps to tell narrative for decisions made.

It is important to record information that can help support the proper treatment plan and reasoning for such services.

PROBLEM STATEMENT

Home First 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, Age, Income (USD), Credit History, Number of Dependents, Co-Applicant, Loan Amount Request and others. 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.

It's a regression problem, given information about the application we have to predict whether the they'll be to pay the loan or not.

We'll start by exploratory data analysis, then pre-processing, and finally we'll be testing different models such as Logistic regression, Support Vector Machine, Random Forest and Decision Trees.

OBJECTIVE

1. Data Analysis is done to scrutinize the given data set and encapsulate their main characteristics.
2. To predict the Accuracy of the dataset, we need to apply Regression algorithm. After training and testing the model the r2_score has to be evaluated for all the algorithm

SYSTEM SPECIFICATION

Hardware Specification:

- **OS:** Windows 10 pro
- **Version:** 10.0.19042 build 19042
- **Processor:** Intel(R) Core (TM) i5-2520M CPU @ 2.50GHz, 2501 MHz, 2 Core(s),4 Logical Processor(s)
- **System Type:** 64-bit operating system, x64-basedprocessor
- **Ram:** 4 gigabytes (GB)
- **Graphics Card:** DirectX 9 or later with WDDM 1.0 driver
- **Display:** 800 x 600
- **Pen and Touch:** No pen or touch input is available for this display.

Software Specification:

- Jupyter Notebook
- Anaconda3
- Google Chrome or Microsoft edge of Latest Version
- Alternatively Google co-lab
- Internet connectivity is to import the libraries required for execution and take advantage of some features.

Libraries:

- Numpy
- Pandas
- Scikit Learn
- Seaborn
- Matplotlib

EXPLORATORY DATA ANALYSIS

1. LOADING THE DATA SET:

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

```
In [3]: data = pd.read_csv("classified-data.csv")
```

```
In [4]: data.head()
```

Out[4]:

	Customer ID	Name	Gender	Age	Income (USD)	Income Stability	Profession	Type of Employment	Location	Am Req (U
0	C-36995	Frederica Shealy	F	56	1933.05	Low	Working	Sales staff	Semi-Urban	7280
1	C-33999	America Calderone	M	32	4952.91	Low	Working	NaN	Semi-Urban	4683
2	C-3770	Rosetta Verne	F	65	988.19	High	Pensioner	NaN	Semi-Urban	4559
3	C-26480	Zoe Chitty	F	65	NaN	High	Pensioner	NaN	Rural	8005
4	C-23459	Afton Venema	F	31	2614.77	Low	Working	High skill tech staff	Semi-Urban	11385

5 rows × 24 columns

2. STATISTICAL MEASURE:

```
In [9]: data.describe()
```

Out[9]:

	Age	Income (USD)	Loan Amount Request (USD)	Current Loan Expenses (USD)	Dependents	Credit Score	
count	30000.000000	2.542400e+04	30000.000000	29828.000000	27507.000000	28297.000000	3000
mean	40.092300	2.630574e+03	88826.333855	400.936876	2.253027	739.885381	
std	16.045129	1.126272e+04	59536.949605	242.545375	0.951162	72.163846	
min	18.000000	3.777000e+02	6048.240000	-999.000000	1.000000	580.000000	
25%	25.000000	1.650457e+03	41177.755000	247.667500	2.000000	681.880000	
50%	40.000000	2.222435e+03	75128.075000	375.205000	2.000000	739.820000	
75%	55.000000	3.090593e+03	119964.605000	521.292500	3.000000	799.120000	
max	65.000000	1.777460e+06	621497.820000	3840.880000	14.000000	896.260000	

EXPLORATORY DATA ANALYSIS

3. NUMBER OF MISSING VALUES IN EACH COLUMN:

```
In [21]: data.isnull().sum()
```

```
Out[21]: Customer ID                      0
         Name                             0
         Gender                          53
         Age                              0
         Income (USD)                  4576
         Income Stability              1683
         Profession                       0
         Type of Employment            7270
         Location                         0
         Loan Amount Request (USD)        0
         Current Loan Expenses (USD)    172
         Expense Type 1                   0
         Expense Type 2                   0
         Dependents                    2493
         Credit Score                  1703
         No. of Defaults                  0
         Has Active Credit Card        1566
         Property ID                      0
         Property Age                  4850
         Property Type                    0
         Property Location              356
         Co-Applicant                     0
         Property Price                   0
         Loan Sanction Amount (USD)     340
         dtype: int64
```

4. CLEANING THE DATA SET:

```
In [12]: data = data.dropna()
```

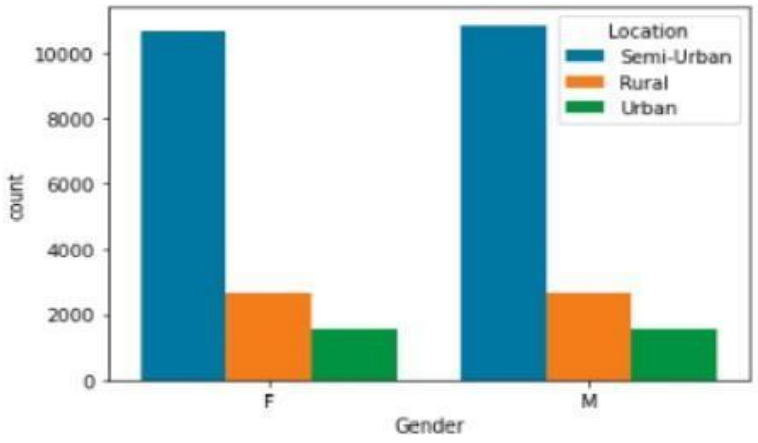
```
In [10]: data.isnull().sum()
```

```
Out[10]: Customer ID                    0
         Name                           0
         Gender                         0
         Age                            0
         Income (USD)                   0
         Income Stability               0
         Profession                     0
         Type of Employment             0
         Location                       0
         Loan Amount Request (USD)      0
         Current Loan Expenses (USD)    0
         Expense Type 1                 0
         Expense Type 2                 0
         Dependents                     0
         Credit Score                   0
         No. of Defaults                0
         Has Active Credit Card         0
         Property ID                    0
         Property Age                   0
         Property Type                  0
         Property Location              0
         Co-Applicant                   0
         Property Price                 0
         Loan Sanction Amount (USD)     0
         dtype: int64
```

DATA VISUALIZATION

1) Count plot between Gender and Location:

```
In [32]: sns.countplot(x='Gender', hue='Location',data=data)
Out[32]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```

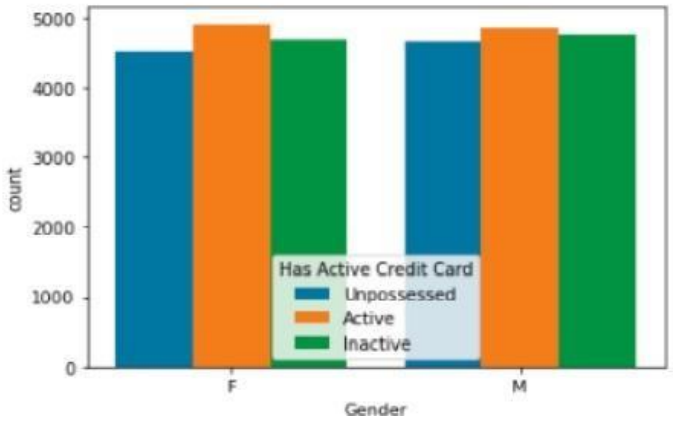


The graph shows the corelation between gender and location, by seeing this graph we can conclude the number of people leaving in different areas.

- More than 10000 people leave in semi-urban areas which include both male and female and the semi–urban area is marked in blue colour for both male and female in the above graph.
- More than 2000 people leave in rural areas which include both male and female and the rural area is marked in orange colour for both male and female in the above graph.
- Less than 2000 people reside in urban areas which include both male and female and urban area is marked in green colour for both male and female in the above graph.

2) Count plot between Gender and Has Active Credit Card:

```
In [26]: sns.countplot(x='Gender', hue='Has Active Credit Card',data=data)
Out[26]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```

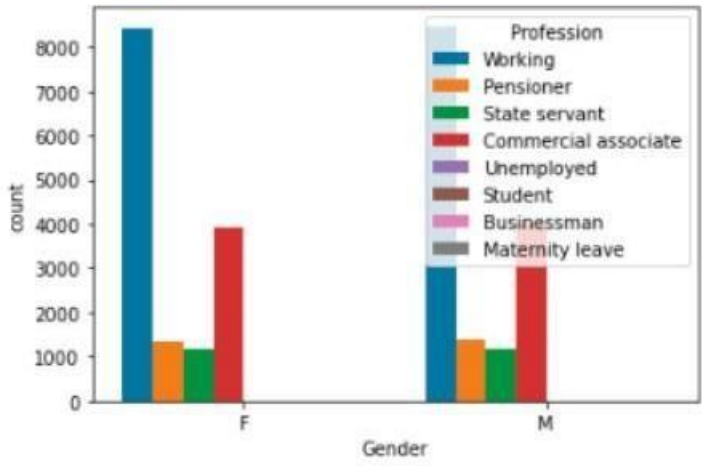


This graph shows the correlation between gender and an active credit card by seeing this graph we can conclude the number of people having an active credit card.

- More than 4000 males and females have unpossessed credit card and it is marked by blue colour in the above graph for both males and females.
- More than 4500 males and females have an active credit card and it is marked by orange colour in the above graph for both males and females.
- More than 4000 males and females have an inactive credit card and it is marked by green colour in the graph for both males and females.

3) Count plot between Gender and Profession:

```
In [34]: sns.countplot(x='Gender',hue='Profession',data=data)
Out[34]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```

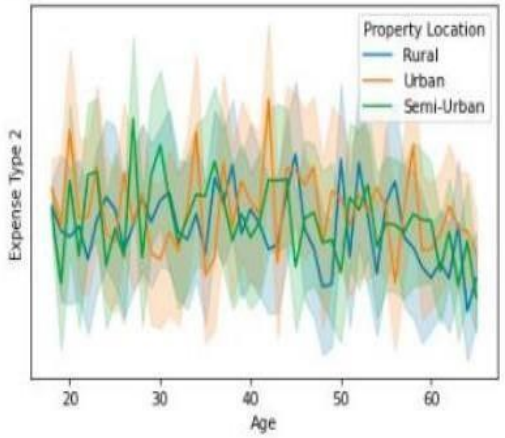


This graph shows the corelation between gender and profession, by seeing this graph we can conclude the profession of individuals based on gender

- More than 8000 members in males and females are working and the number of working people aremarked in blue colour for both males and females in the above graph.
- More than 1000 members in males and females are pensioners and the pensioners are marked orange colour for both males and females in the above graph.
- Around 1000 members are state servants in both males and females and the number of state servants are marked in green colour for both males and females in the above graph.
- Around 4000 members are commercial associates in both males and females and the number of commercial servants are marked in red colour in the above graph.

4) Line plot between Age and Expense Type2:

```
In [60]: ax=sns.lineplot(x='Age', y='Expense Type 2',hue='Property Location', data=data)
```

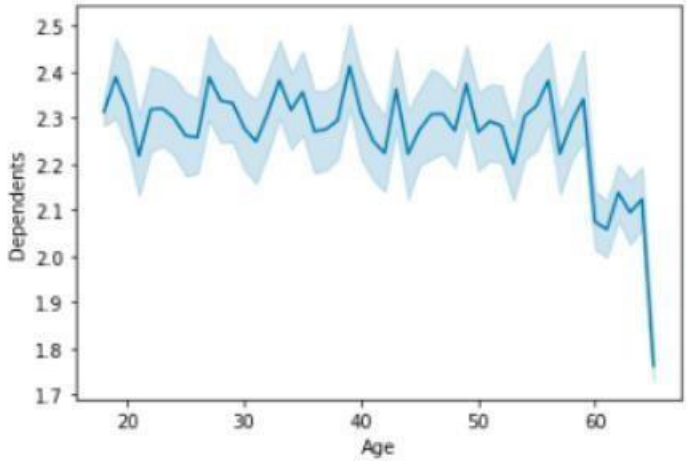


This graph shows the corelation between age and property location

- The maximum number people from the age gap between 40-45 owned property in urban areas and it is indicated in yellow colour in the above graph.
- The maximum number of people from the age gap between 45-55 owned property in rural areas and it is indicated in blue colour in the above graph.
- The maximum number of people from the age gap between 25-30 owned property in semi-urban areas and it is indicated in green colour in the above graph.

5) Line plot between Age and Dependents:

```
In [49]: ax=sns.lineplot(x='Age', y='Dependents', data=data)
```



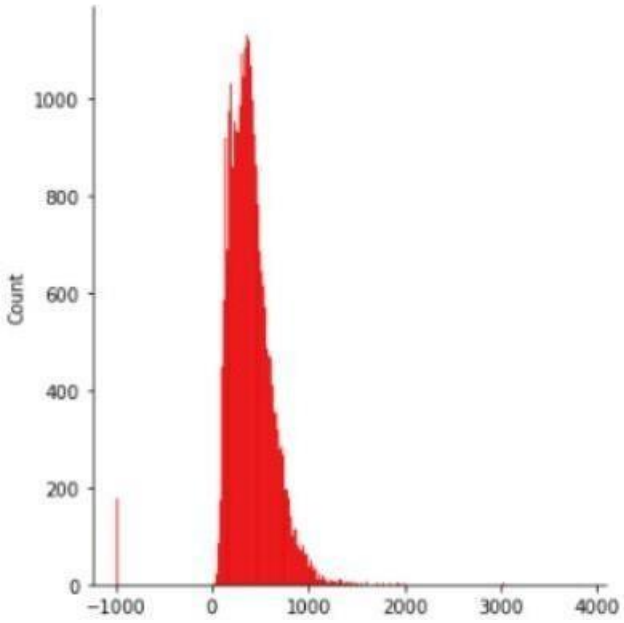
This graph shows the corelation between the age and dependents

- More number of dependents are seen at the age of 20, which isindicated in blue colour in the above graph.
- Minimum number of dependents are seen at the age of 40, which is indicated in blue colour in the above graph.
- The least number of dependents are seen at age above 60, which is indicated in blue colour in the above graph.

6) Dis plot of Count and Current Loan Expenses:

```
In [67]: x = data['Current Loan Expenses (USD)'].values
         sns.displot(x, color = 'red')

Out[67]: <seaborn.axisgrid.FacetGrid at 0x198c4d53cd0>
```

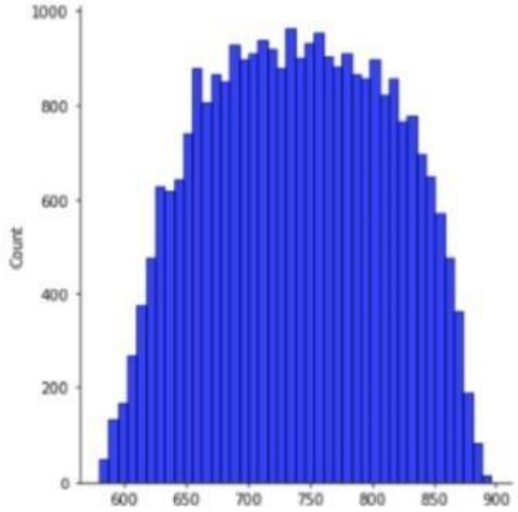


This graph shows corelation between current Loan expenses versus count.

- Maximum loan expenses have been taken from more than 1000 people and it is marked in red colour in the above graph.

7) Dis plot of Count and Credit Score:

```
In [61]: x = data['Credit Score'].values
         sns.displot(x, color = 'blue')
Out[61]: <seaborn.axisgrid.FacetGrid at 0x198c49fc820>
```

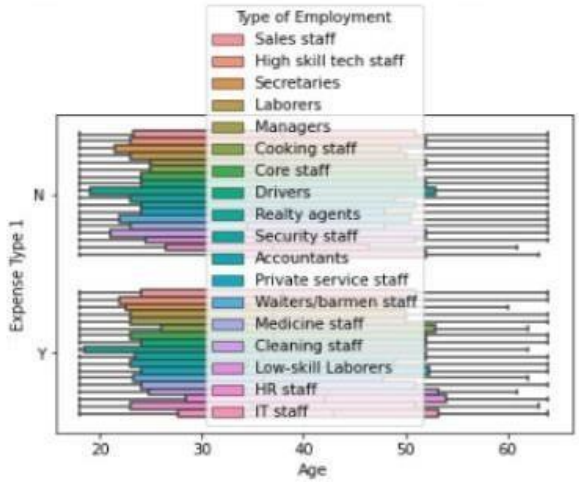


This graph shows the corelation between credit score versus count.

- Approximately around 1000 members have gained the maximum credit score in the above graph and it is indicated in blue colour in the above graph.
- Around 800 members have gained the minimum credit score in the above graph.
- Very less members possess very least credit score in the above graph.

8) Boxplot between Age and Expense Type 1:

```
In [78]: ax = sns.boxplot(x='Age',y='Expense Type 1',hue='Type of Employment',data=data)
```

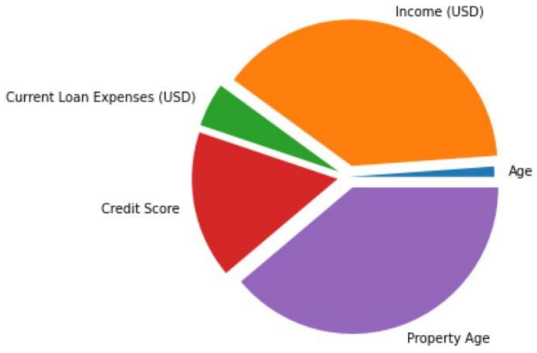


This graph shows the corelation between age versus expense type 1.

- Expense type 1 includes yes or no.
- In no section reality agents have very more expenses between the age gap of 20-5o and above it is indicated in green colour in the above graph.
- In no section HR staffs have very less expenses between the age gap of 25-45 it is indicated in pink colour in the above graph.
- In yes section the HR staffs have more expenses between the age gap of 25-45 it is indicated in pink colour in the above graph.
- In yes section the waiters/barmen staff have very less expenses between the age gap of 25-45 it is indicated in blue colour in the above graph.

9) Pie chart:

```
In [82]:  1 plt.pie(exp_vals,labels=exp_labels,radius=1.3,explode=[0,0.1,0.1,0.1,0.1])
          2 plt.show()
```



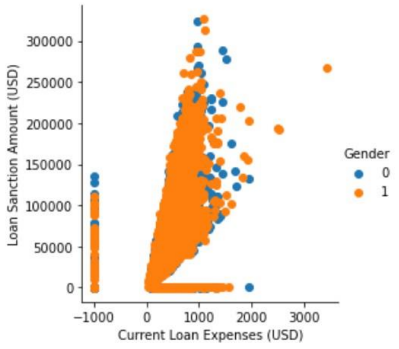
Here the Pie chart is divided into sections to represent values of different

Sizes.

- The above pie chart is constructed for the main attributes like age, income Current loan expenses, property age and credit score.
- The pie chart is constructed using the exact values given and divided up into segments which represent each value.

10) Scatter plot between Loan Sanction Expenses (USD) and Current Loan Expenses (USD):

```
In [75]:
# Using SCATTER REPRESENTATION
# Imported libraries... x-axis: loan_status, y_axis: Loan_Amount and representing in terms of Gender_Sectio
sns.FacetGrid(data,hue="Gender",size=4) \
.map(plt.scatter,"Current Loan Expenses (USD)","Loan Sanction Amount (USD)") \
.add_legend()
plt.show()
```



The above scatter representation shows the relation between loan sanction amount (USD) and current loan expenses (USD)

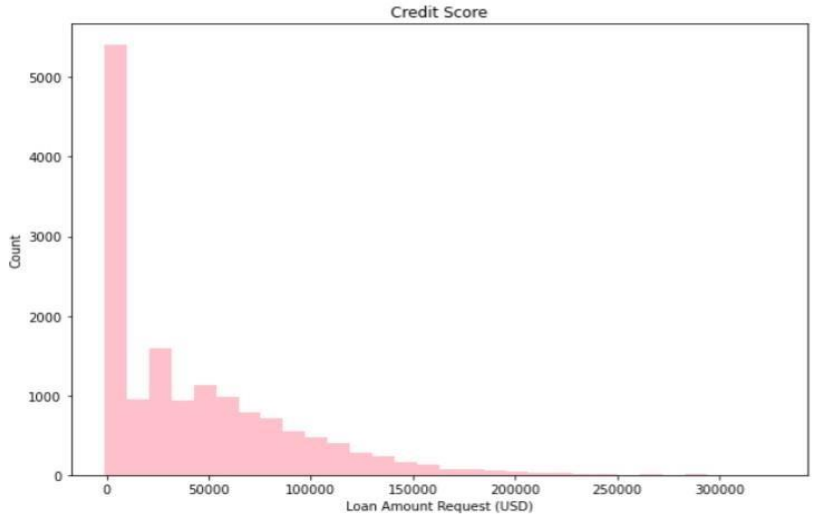
- Here the graph is linearly proportional.
- As the loan sanction amount increases (USD), current loan expenses (USD) also increase.
- The male is having large current loan expenses when compared tothe female.
- All the data sets seem to be concentrated within 100000-180000 of loan sanction amount (USD) and within 100-1200 of current loan expenses (USD).
- We can also observe that there is slight decrease in the loan sanction amount (USD) above 180000(USD)

11) Histogram between Count and Loan Amount Request (USD):

```
In [69]: # Using HISTOGRAM REPRESENTATION
         # x-axis: Loan figures, y_axis: count, Title: Loan taken by Customers

         plt.figure(figsize = (10,7))
         x = data["Loan Sanction Amount (USD)"]
         plt.hist(x, bins = 30, color = "pink")
         plt.title("Credit Score")
         plt.xlabel("Loan Amount Request (USD)")
         plt.ylabel("Count")

Out[69]: Text(0, 0.5, 'Count')
```

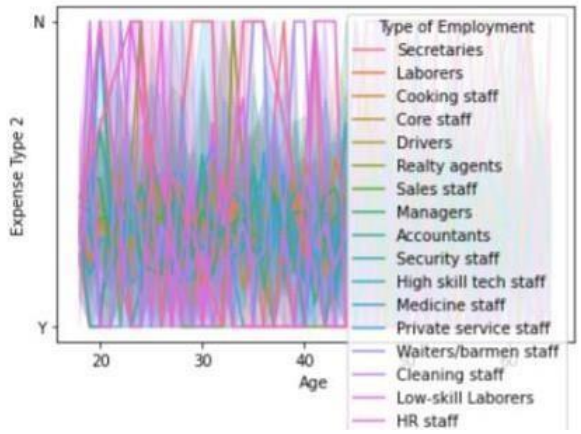


The Histogram is used to summarize discrete or continuous data that are measured on an interval scale.

- Here the histogram represents the relation between count and Loan amount requested (USD)
- From the above histogram we can observe that the loan amount request (USD) ranges from 0-100000 for the count 0-5000.
- We can also interpret that around 1500 count are requesting an average amount of 25000 USD which is quite highest when compared to others.
- The loan amount request (USD) above 100000 USD is very less.

12) Line plot:

```
In [35]: ax=sns.lineplot(x='Age', y='Expense Type 2',hue='Type of Employment', data=data)
```



The above line plot shows the relation between the age and expense type.

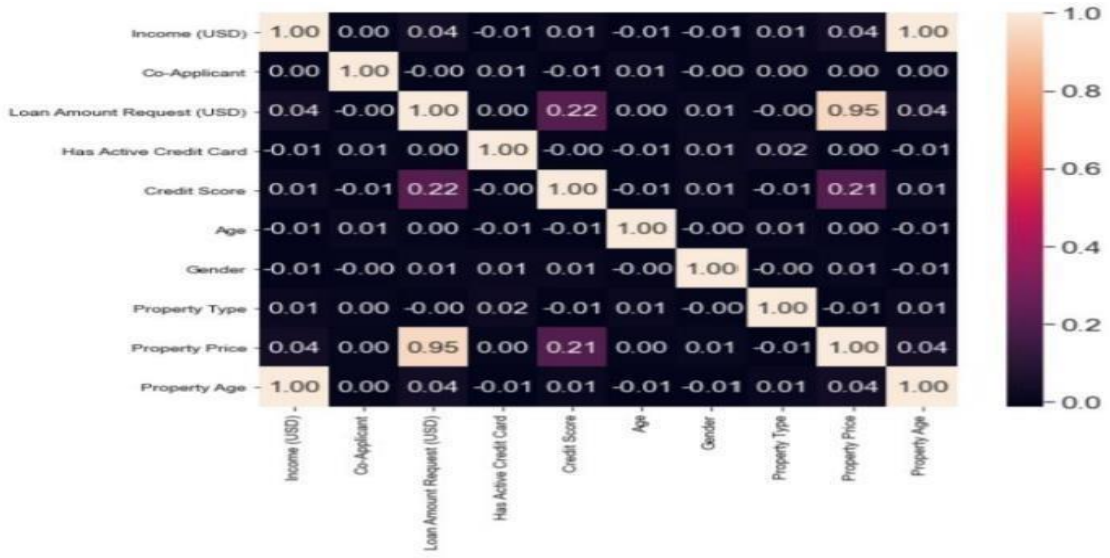
- Here the type of employment is also shown as hue.
- We can depict that the HR staff, low skill laborers and security staff have the high expenses when compared to others.

13) Heat map:

```
In [84]:
# Correlation Matrix of the columns given below

cols = ['Income (USD)','Co-Applicant','Loan Amount Request (USD)','Has Active Credit Card','Credit Score'
        'Gender','Property Type','Property Price','Property Age']
f, ax = plt.subplots(figsize=(10, 7))
cm = np.corrcoef(data[cols].values.T)
sns.set(font_scale=1.5)
hm = sns.heatmap(cm,
                 cbar=True,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=cols,
                 xticklabels=cols)

plt.show()
```



A Heat map is a two-dimensional representation of data in which values are represented by colours. Heat map allow users to understand and analyse complex data sets.

- The above graph is the heatmap of the loan approval prediction, which shows the relation between the two variables in the dataset, one plotted on each axis.
- The variation in colour may be due to hue or intensity.
- The diagonals are all one because those squares are correlating each variable to itself. (So, it’s a perfect correlation).

1) LINEAR REGRESSION:

In [101]:
```
X = data[['Gender' , 'Age', 'Expense Type 1', 'Expense Type 2', 'Income (USD)', 'Loan Amount Request (USD)'
```

In [102]:
```
y = data[['Property Price', 'Property Age', 'Property Location']]
```

In [103]:
```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train ,y_train)
```

Out[103]:
```
LinearRegression()
```

In [104]:
```
X_test
```

Out[104]:
```
array([[1.0000000e+00, 5.7000000e+01, 2.0000000e+00, ..., 1.5124005e+05,
        1.0586803e+05, 4.9748000e+02],
       [0.0000000e+00, 5.7000000e+01, 2.0000000e+00, ..., 9.2496360e+04,
        0.0000000e+00, 5.2058000e+02],
       [0.0000000e+00, 2.7000000e+01, 1.0000000e+00, ..., 3.6306510e+04,
        2.3599230e+04, 3.7891000e+02],
       ...,
       [1.0000000e+00, 3.2000000e+01, 5.0000000e+00, ..., 6.8448100e+04,
        5.1336080e+04, 4.3765000e+02],
       [0.0000000e+00, 5.3000000e+01, 3.0000000e+00, ..., 4.0863620e+04,
        2.8604530e+04, 2.1037000e+02],
       [1.0000000e+00, 2.0000000e+01, 4.0000000e+00, ..., 3.7328670e+04,
        2.6130070e+04, 1.9330000e+02]])
```

In [105]:
```
clf.predict(X_test)
```

Out[105]:
```
array([[ 1.00000000e+00],
       [ 2.71379783e-14],
       [ 1.00000000e+00],
       ...,
       [ 1.00000000e+00],
       [-2.61592545e-14],
       [-8.05556112e-15]])
```

In [106]:
```
y_test
```

Out[106]:
```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [0],
       [0]], dtype=int64)
```

In [107]:
```
clf.score(X_test ,y_test)
```

Out[107]:
```
1.0
```

In [108]:
```
clf.score(X_train,y_train)
```

Out[108]:
```
1.0
```

In [109]:
```
# Y contains all the outputs and X contains all the inputs. We will test on the machine if it gives to expe
# corresponding Inputs.
expected = y_test
predicted = model.predict(X_test)
```

In [110]:
```
# Importing Libraries and class
from sklearn import metrics
```

In [111]:
```
print(metrics.classification_report(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2488
           1       1.00      1.00      1.00      5037

    accuracy                           1.00      7525
   macro avg       1.00      1.00      1.00      7525
weighted avg       1.00      1.00      1.00      7525
```

In [112]:
```
print(metrics.confusion_matrix(expected, predicted))
```

```
[[2488    0]
 [   0 5037]]
```

2) RANDOM FOREST:

In [154]:
```
#importing classes and libraries

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```

In [155]:
```
model.fit(X_train,y_train)
```

Out[155]:
```
RandomForestClassifier()
```

In [156]:
```
model.score(X_train,y_train)
```

Out[156]:
```
1.0
```

In [157]:
```
model.score(X_test,y_test)
```

Out[157]:
```
1.0
```

In [158]:
```
# Y contains all the outputs and X contains all the input.we will test on the machine if it gives to expecte
# corresponding Inputs

expected = y_test
predicted = model.predict(X_test)
```

In [159]:
```
#generating Report

print(metrics.classification_report(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1480
           1       1.00      1.00      1.00      3035

    accuracy                           1.00      4515
   macro avg       1.00      1.00      1.00      4515
weighted avg       1.00      1.00      1.00      4515
```

In [160]:
```
# Output in the form of matrix

print(metrics.confusion_matrix(expected,predicted))
```

```
[[1480    0]
 [   0 3035]]
```

3) DECISION TREE:

In [161]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

In [162]:
```
model.fit(X_train,y_train)
```

Out[162]:
```
DecisionTreeClassifier()
```

In [163]:
```
model.score(X_train,y_train)
```

Out[163]:
```
1.0
```

In [164]:
```
model.score(X_test,y_test)
```

Out[164]:
```
1.0
```

In [165]:
```
expected = y_test
predicted = model.predict(X_test)
```

In [166]:
```
print(metrics.classification_report(expected,predicted))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1480
           1       1.00      1.00      1.00      3035

    accuracy                           1.00      4515
   macro avg       1.00      1.00      1.00      4515
weighted avg       1.00      1.00      1.00      4515
```

In [167]:
```
print(metrics.confusion_matrix(expected,predicted))
```

```
[[1480    0]
 [   0 3035]]
```

Random state argument

In [168]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.5,random_state=10)
X_test
```

Out[168]:
```
array([[1.0000000e+00, 5.7000000e+01, 2.0000000e+00, ..., 1.5124005e+05,
        1.0586803e+05, 4.9748000e+02],
       [0.0000000e+00, 5.7000000e+01, 2.0000000e+00, ..., 9.2496360e+04,
        0.0000000e+00, 5.2058000e+02],
       [0.0000000e+00, 2.7000000e+01, 1.0000000e+00, ..., 3.6306510e+04,
        2.3599230e+04, 3.7891000e+02],
       ...,
       [1.0000000e+00, 3.2000000e+01, 5.0000000e+00, ..., 6.8448100e+04,
        5.1336080e+04, 4.3765000e+02],
       [0.0000000e+00, 5.3000000e+01, 3.0000000e+00, ..., 4.0863620e+04,
        2.8604530e+04, 2.1037000e+02],
       [1.0000000e+00, 2.0000000e+01, 4.0000000e+00, ..., 3.7328670e+04,
        2.6130070e+04, 1.9330000e+02]])
```

4) POLYNOMIAL REGRESSION:

In [250]:
```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

In [251]:
```
X,y=load_boston(return_X_y=True)
poly = PolynomialFeatures(degree = 2)
X = poly.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5)
```

In [252]:
```
model = LinearRegression()
```

In [253]:
```
model.fit(X_train, y_train)
expected_y  = y_test
predicted_y = model.predict(X_test)
```

In [254]:
```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.936248611544911
0.7050760987307356
```

We are looking for relation between x ='Age' and y= ' Dependents' using Polynomial Regression

In [265]:
```
from sklearn.linear_model import LinearRegression
```

In [256]:
```
x=data['Age'].values
y=data['Dependents'].values
```

In [257]:
```
x = x.reshape(-1,1)
```

In [258]:
```
poly = PolynomialFeatures(degree=10)
```

In [259]:
```
X_poly = poly.fit_transform(x)
```

In [260]:
```
poly.fit(X_poly,y)
```

Out[260]:
```
PolynomialFeatures(degree=10)
```

In [261]:
```
linreg = LinearRegression()
```

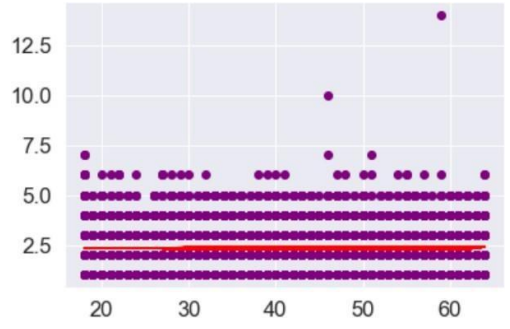
In [262]:
```
linreg.fit(X_poly,y)
```

Out[262]:
```
LinearRegression()
```

In [263]:
```
y_pred =linreg.predict(X_poly)
```

In [264]:
```
plt.scatter(x,y,color='purple')
plt.plot(x,y_pred, color='red')
```

Out[264]:
```
[<matplotlib.lines.Line2D at 0x1b96927c880>]
```



5) LOGISTIC REGRESSION:

In [135]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [136]:
```
X=data[['Gender','Age','Dependents','Credit Score','Property Type','Property Age','Property Price','Property

'Co-Applicant','Income (USD)','Loan Amount Request (USD)','Loan Sanction Amount (USD)',
'Current Loan Expenses (USD)']].values

y=data[["Expense Type 2"]].values
```

In [137]:
```
# Importing Libraries and classes
# Dividing the data in 7:3 Ratio for Training and Testing respectively

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

In [138]:
```
# Training the Model

model.fit(X_train,y_train)
```

Out[138]:
```
LogisticRegression()
```

In [139]:
```
model.score(X_test, y_test)
```

Out[139]:
```
0.6722037652270211
```

In [140]:
```
model.score(X_train,y_train)
```

Out[140]:
```
0.6780256288561937
```

In [141]:
```
# Y contains all the outputs and X contains all the inputs. We will test on the machine if it gives to expe
# corresponding Inputs.

expected = y_test
predicted = model.predict(X_test)
```

In [142]:
```
# Importing Libraries and class

from sklearn import metrics
```

In [143]:
```
print(metrics.classification_report(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1480
           1       0.67      1.00      0.80      3035

    accuracy                           0.67      4515
   macro avg       0.34      0.50      0.40      4515
weighted avg       0.45      0.67      0.54      4515
```

In [144]:
```
print(metrics.confusion_matrix(expected, predicted))
```

```
[[   0 1480]
 [   0 3035]]
```

6) SUPPORT VECTOR MACHINE:

In [145]:
```
#Importing class nad libraries

from sklearn.svm import SVC
model = SVC()
```

In [148]:
```
model.fit(X_train,y_train)
```

Out[148]:
```
SVC()
```

In [149]:
```
model.score(X_train,y_train)
```

Out[149]:
```
0.6781205505457997
```

In [150]:
```
model.score(X_test,y_test)
```

Out[150]:
```
0.6722037652270211
```

In [151]:
```
from sklearn import metrics
```

In [152]:
```
#Obtaining Report

print(metrics.classification_report(expected, predicted))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1480
           1       0.67      1.00      0.80      3035

    accuracy                           0.67      4515
   macro avg       0.34      0.50      0.40      4515
weighted avg       0.45      0.67      0.54      4515
```

In [153]:
```
#Output in the form

print(metrics.confusion_matrix(expected,predicted))
```

```
[[   0 1480]
 [   0 3035]]
```

ML MODEL CHART

Sl Number	Algorithm Name	R2_score
01	**Linear Regression Model**	**1.0**
02	**Random Forest**	**1.0**
03	**Decision Tree**	**1.0**
04	**Polynomial Regression**	**0.705076**
05	**Logistic Regression Model**	**0.678025**
06	**Support Vector Machine**	**0.672203**

HURDLES

Beginning, we faced issue in cleaning the given data set, to identify and remove the null values and replace them with appropriate values. To overcome this, we used Pandas inbuilt functions to replace the null values. Then we replaced all the null values with mean of that specific column.

Next, we addressed issue in predicting x and y values , we overcame by splitting our original dataset into input(X) and output(Y) columns, then call the function passing both arrays and split correctly as train and test subsets.

At last, we faced issue in selecting the best graphs as there were many graphs. We overcame this by executing the correct code and choosing the graphs with correct logic. We preferred the charts type that fits the size of our data best and representing it clearly without being bestrewed.

CONCLUSION

Lately people depend on bank loans to meet their wishes. The fee of loan packages will increase with a rapid speed in current years. There is need of automation of this system so that loan approval is much less risky.

In this project some ML algorithms like Linear Regression, Logistic regression, Decision Tree, Random Forest, Support Vector Machine, etc. are implemented to expect the loan approval for customers. We further formulated a predictive model using Linear Regression that composed of the most important features for predicting customers credit worthiness. We've gone through a good portion of the data science in this project namely EDA and Modelling.

As it certainly a very important procedure for a bank to check whether an applicant, applying for a term loan should be approved or not. This project mainly focuses on making the tedious task mechanized. All the required algorithms were implemented successfully and accurate results were generated.

BIBLIOGRAPHY

➢ Books referred:

- Loan Approval Predictor using Data Science and Machine Learning by Shail Vatsal Vashist (161480). Under the supervision of Dr. Aman Sharma, Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh.
- Loan Default Prediction using Supervised Machine Learning Algorithms by Daria Granstrom. Under the supervision of Johan Abrahamsson, KTH Royal Institute of Technology School of Engineering Sciences, Stockholm, Sweden- 2019.

➢ Article referred:

- Predict Loan Approval in Banking System Machine Learning Approach for Cooperative Banks Loan Approval by Amrutha S. Aphale, Dr. Sandeep R. Shinde. Paper ID: IJERTV9IS080309, Volume & Issue: Volume 09, issue 08 (August 2020), Publisher Name: IJERT.

➢ International Research journal referred:

- Prediction for Loan Approval using Machine Learning Algorithm by Ashwini S. Kadam, Ankitha A. Aher, Shraddha R. Nikim, Gayatri V. Shelke, Yeola, Maharashtra, India.

➢ Websites referred:

- **www.youtube.com**
- **www.slideshare.net**
- **www.coursera.org**
- **www.encyclopedia.com**
- **www.geeksforgeeks.com**
- **https://cognitiveclass.ai**