Loan prediction report

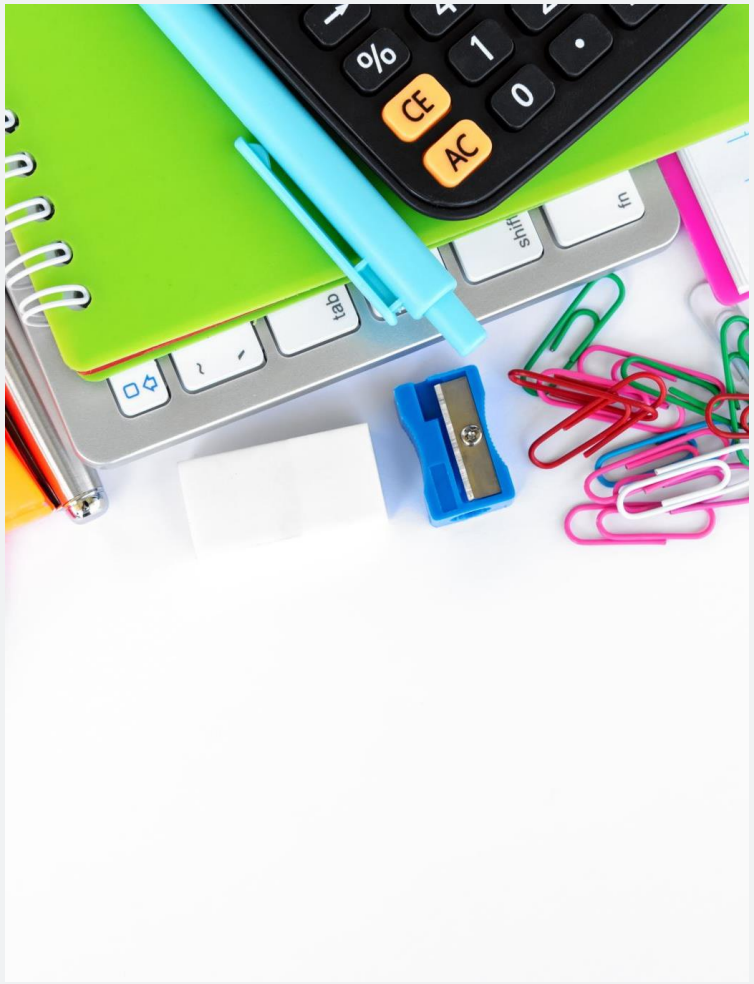
Dr :Verene Nashaat

Introduction

- Loans are important to banks so companies want to automate the loan eligibility process based on customer details provided while filling an application form and these details like Gender , loan_id , property _area ,material-status and so ….on so, the loan companies loan after an accurate process of validation. To automate this process companies have a big problem knowing customers' segments those are eligible for loan amount.

This report is based on a model created to predict if the user can take a loan or not . This report basically includes the steps followed during the implementation of machine learning.

Description :



Overview of the Data

.

The first step is to look at the data we're working with. Realistically , most of the data we will get can have errors , and it's important to identify these errors before analyzing data. So ,we should start by reading the data using the function **read.csv().**

And before we analyze the data , we should make a smart object and we need to identify independent variables and the dependent variable (target) here is **Loan-status**.

Clean the data and make it ready to use as we have many errors in the dataset, we must fix them like there are missing values or invalid values in some variables and to clean it we need to **import pandas as pd** and **import numpy as np.**

And since we cannot enter string values to model so we used the function label encoder .

To clean data.

In this dataset, we'll assume that the missing values are systematic because the missing data are coming in certain variables in random manner .Also ,we note that missing values are on both numerical and categorial data.

we filled all the numeric values with **mean** and to fill the categorial values with **mode** and the mode is the most frequent value in the columns .then we used label encoder as we can see here in this picture . This is the way we used to clean the data .And there is another way to clean the data is to drop all the nulls we have and then do label encoding. But we noticed that when we used the mean and mode , we got a higher accuracy than in a drop of null values.

```
le = LabelEncoder()

o = df["Gender"].mode()
df["Gender"].fillna(o[0], inplace=True)
m = df["Married"].mode()
df["Married"].fillna(m[0], inplace=True)
s = df['Self_Employed'].mode()
df['Self_Employed'].fillna(s[0], inplace=True)
x = df["LoanAmount"].mean()
df["LoanAmount"].fillna(x, inplace=True)
lmt = df["Loan_Amount_Term"].mean()
df["Loan_Amount_Term"].fillna(lmt, inplace=True)
ch = df["Credit_History"].mode()
df["Credit_History"].fillna(ch[0], inplace=True)
h = df["Dependents"].mode()
df["Dependents"].fillna(h[0], inplace=True)
# print(df.isnull().sum())
df["Gender"] = le.fit_transform(df["Gender"])
df["Married"] = le.fit_transform(df["Married"])
df["Education"] = le.fit_transform(df["Education"])
df["Self_Employed"] = le.fit_transform(df["Self_Employed"])
df["Loan_Status"] = le.fit_transform(df["Loan_Status"])
df["Property_Area"] = le.fit_transform(df["Property_Area"
df["Dependents"] = le.fit_transform(df["Dependents"])
df["Loan_ID"] = le.fit_transform(df["Loan_ID"])
```

Building predictive Models.

Click to add text

Now it's time to analysis our data which is splitting the data into training and test sets.

A training set is the subset of the data that we want to use to train our models but the test set is a random subset of the data which are derived from the training set. We will use the test set to validate our models as un-foreseen data.

In our data , it's easy to overfit the data. Overfit in simple terms means that the model will learn the training set that it won't be able to handle most of the cases it has never seen before. Therefore , we are going to score the data using our test set. Once we split the data , we will treat the testing set like it no longer exists.

And to split into train and test we made the random-state =8 and for test we split it 10% and that’s the best we've come up with.

Logistic Regression.

We will now start with our first algorithm .As here we want to classify between the people who have taken loan or not. we have used the logistic regression .The purpose of this algorithm to find a plane that separates 2 types. Y variable belongs to 1 or 0.in 2_d we have to figure out a line that exactly separates two classes. **Here** is the code for logistic regression. We notice that when the max-iter =300 and random-state=8 the accuracy is 0.91 (best accuracy) , But if we changed these numbers and made max-iter = 50 ,random-state=7 ,we noticed the accuracy was lower and it's not the best for our model.

```
tic_Regression():
LogisticRegression(solver='lbfgs', max_iter=300)
 the model with data
t(x_train, y_train)
dict with test dataset
dict_LR = lr.predict(x_test)
nt(y_predict_LR)
acy_Score_LR = metrics.accuracy_score(y_test, y_predic
("*****************")
("Logistic_Regression accuracy :", accuracy_Score_LR)
ssification_report
()
("*classification_report*")
(classification_report(y_test, y_predict_LR))
("*****************")
```

Decision Tree.

```
def DecisionTree():
    model = DecisionTreeClassifier(random_state=0, max_depth=3)
    # fit the model with data
    model.fit(x_train, y_train)
    # x=model.feature_importances_
    # print(x)
    # predict with test dataset
    y_predict_DecisionTree = model.predict(x_test)
    # print(y_predict_DecisionTree)
    accuracy_Score_DecisionTree = metrics.accuracy_score(y_test, y_predict
    print("*****************")
    print("DecisionTree accuracy :", accuracy_Score_DecisionTree)
    # classification_report
    print()
    print("*classification_report*")
    print(classification_report(y_test, y_predict_DecisionTree))
    print("*****************")
```

This is supervised machine learning algorithm mostly used for classification problems. This model uses a different algorithm to split a node into two or more sub-nodes. **Here** is the code of the decision tree. We made the random-state =0 and max-depth=5 we got an accuracy =0.88 but when max-depth =3 the accuracy=0.91 and that's the best we've come up with.

```
def supportVectorMachine():
    cl = SVC(kernel='rbf')
    # fit the model with data
    cl.fit(x_train, y_train)
    # predict with test dataset
    y_predict_SVM = cl.predict(x_test)
    # print(y_predict_SVC)
    accuracy_Score_SVM = metrics.accuracy_score(y_test, y_pre
    print("*****************")
    print("supportVectorMachine accuracy :", accuracy_Score_S
    # classification_report
    print()
    print("*classification_report*")
    print(classification_report(y_test, y_predict_SVM))
```

Support vector machine.

This is supervised learning model with associated learning algorithm that analyze the data for classification and regression analysis. It's a discriminative classifier that is formally designed by separative hyperplane. We used the svm kernels ,redial basic function (rbf) to get a high accuracy and that accuracy is 0.75 .

Random forest.

This is tree based ensemble model which helps in improving the accuracy of the model.it combines a large of decision trees to build a powerful predicting model. when we tried n-estimators = 19 and random-state =3 , we got accuracy =0.88.But when we tried n-estimators =21 and random-state =0 , the accuracy was 0.90 and that's the best we've come up with.

```
f RandomForest():
  models = RandomForestClassifier(
      n_estimators=21, criterion='entropy', random_state=0)
  models.fit(x_train, y_train)
  # importance = models.feature_importances_
  # for i in range (xtrain.shape[]):
  # print (name[i],importance[i])
  # predict with test dataset
  y_predict_random = models.predict(x_test)
  # print(y_predict_DecisionTree)
  accuracy_Score_random = metrics.accuracy_score(
      y_test, y_predict_random)
  print("*****************")
  print("RandomForest accuracy :", accuracy_Score_random)
  # classification_report
  print()
  print("*classification_report*")
  print(classification_report(y_test, y_predict_random))
```

Navie Bayes.

```
def NaiveBayes():
    # train the model using NaiveBayes
    gnb = GaussianNB()
    # fit the model with data
    gnb.fit(x_train, y_train)
    # predict with test dataset
    y_predict_NaiveBayes = gnb.predict(x_test)
    # print (y_predict)
    NaiveBayes_Accuracy = metrics.accuracy_score(y_test, y_predict_Naivel
    print("*****************")
    print("Naive Bayes model accuracy :", NaiveBayes_Accuracy)
    # classification_report
    print()
    print("*classification_report*")
    print(classification_report(y_test, y_predict_NaiveBayes))
```

This is a machine learning model that is used for large volumes of data. It is a probabilistic classifier , which means it predicts on the basic of the probability of an object. **Here** is the code of navie bayes.
The accuracy in this model is 0.88 and this the best we've come up with.

K-Nearest Neighbour.

```
def KNN():
    # Fitting K-NN classifier to the training set
    classifier = KNeighborsClassifier(n_neighbors=40, metric='mi
    classifier.fit(x_train, y_train)
    # Predicting the test set result4

    y_pred_KNN = classifier.predict(x_test)
    KNN_Accuracy = metrics.accuracy_score(y_test, y_pred_KNN)
    print("*****************")
    print("KNN model accuracy :", KNN_Accuracy)
    # classification_report
    print()
    print("*classification_report*")
    print(classification_report(y_test, y_pred_KNN))
```

KNN is a supervised machine learning algorithm that can solve both classification and regression problem statements. The number of nearest neighbors to a new unknown variable that has to be predicted is denoted by the symbol 'K'.

Here is the code of the knn.

So , when we made the n-neighbors =30 , the accuracy was 0.74 .But when we put it 40 , the accuracy was 0.75 and that's best we've come up with.

Stander Scaler.

```
ef standardScaler():
    scaler = StandardScaler(copy=True, with_
    columns = ['Loan_ID', 'Dependents', 'App
    'CoapplicantIncome', 'LoanAmount', 'Loan
    x = df[columns]
    x = scaler.fit_transform(x)
    print("***")
    print('x\n', x[:100])
    print('y\n', y[:60])
```

It is a function that standardize the data values into standard format. We initially create an object of the stander scaler () function. Further ,we use **fit -transform ()** along with assigned object to transform the data and standardize it. As we can see the code here.

Classification report

- It is a performance evaluation metric in machine learning.
- It is used to show the precision , recall , f1 score, and support of your trained classification model.
- To view the classification report of machine learning model , we must first train a machine learning model and that's what we have done in previous pages . So here is the code to view classification report .

```
19  from sklearn.metrics import classification_report
20  print(classification_report(y_test, predictions))
```

Feature Selection.

```
# Feature selection
sel = SelectFromModel(model)
sel.fit(x, y)
selected_features = sel.transform(x)
print(sel.get_support())
print(df.shape)
print(selected_features.shape)
```

- Feature selection is the method of reducing the input variables to your model by using only relevant and getting red of noise in data.
- So ,we take an object from the function select-from-model and select the model . Here is the code of feature selection.

Feature extraction.

Click to add text

- It refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. And since our data is supervised we used the linear discriminant analysis (lda) is used to reduce the number of the features to a more manageable number. And it has a higher score than other algorithms. So here is the code of it.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
def lda():
    lda_model = LDA(n_components=1)
    X_train_lda = lda_model.fit_transform(x_train, y_train)
    X_test_lda = lda_model.transform(x_test)
    print( lda_model.score(x_train, y_train))
```

Here is teams' members :

ID	Name
• 20201700412	شيماء حسين محمد السيد
• 20201700264	رقيه عبد السميع محمود ابراهيم
• 20201701145	منه الله محمد علي محمد
• 20201700209	جهاد مصطفي عبد الرازق
• 20201700960	هبه محروس رمضان محروس
• 20201700208	جهاد محمدي حسن السيد
• 20201701115	كريمه صبجي ابو العلا عبد الخالق