Loan Prediction Automation

CSC 2302: Data Science/Machine Learning Project

Laura Thornton
Vijay Kalavakolanu
Zelin (Victor) Wang

Problem

- Automate loan approval process for applicants based on certain information provided via form

Data: Variables

Loan_ID	- Unique identifier (LP00000)
Gender	- (Male/Female)
Married	- Marital status (Y/N)
Dependents	- # of dependents (0/1/2/3+)
Education	- (Graduate/Not Graduate)
Self_Employed	- Self-employment status (Y/N)
ApplicantIncome	- Applicant's Income ($)
CoapplicantIncome	- Coapplicant's Income, if applicable ($)
LoanAmount	- Loan amount in thousands ($)
Loan_Amount_Term	- Term of loan (# months)
Credit_History	- Credit History (0/1.0)
Property_Area	- Area (Urban/Semiurban/Rural)
Loan_Status	- Loan approved or not (Y/N)

Data: Pre-Processing

Step 1: Identify categorical and continuous/numerical variables

- Loan_ID - Unique identifier (LP00000)
- Gender - (Male/Female)
- Married - Marital status (Y/N)
- Dependents - # of dependents (0/1/2/3+)
- Education - (Graduate/Not Graduate)
- Self_Employed - Self-employment status (Y/N)
- ApplicantIncome - Applicant's Income ($)
- CoapplicantIncome - Coapplicant's Income, if applicable ($)
- LoanAmount - Loan amount in thousands ($)
- Loan_Amount_Term - Term of loan (# months)
- Credit_History - Credit History (0/1.0)
- Property_Area - Area (Urban/Semiurban/Rural)
- Loan_Status - Loan approved or not (Y/N) - only in train data

Data: Pre-Processing

Step 2: Import data using pandas and look for trends between categorical variables and loan status

```
Probability of loan approval based on gender:
        Loan_Status
Gender
Female     0.669643
Male       0.693252
```

```
Probability of loan approval based on number of dependents:
            Loan_Status
Dependents
0              0.689855
1              0.647059
2              0.752475
3+             0.647059
```

```
Probability of loan approval based on self employment:
               Loan_Status
Self_Employed
No                0.686000
Yes               0.682927
```

```
Probability of loan approval based on property area:
               Loan_Status
Property_Area
Rural             0.614525
Semiurban         0.768240
Urban             0.658416
```

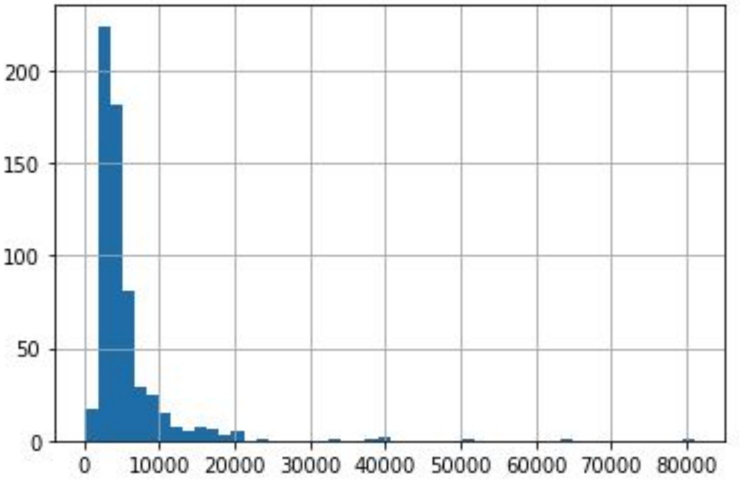
```
Probability of loan approval based on marital status:
         Loan_Status
Married
No          0.629108
Yes         0.716080
```

```
Probability of loan approval based on credit history:
                Loan_Status
Credit_History
0.0                0.078652
1.0                0.795789
```

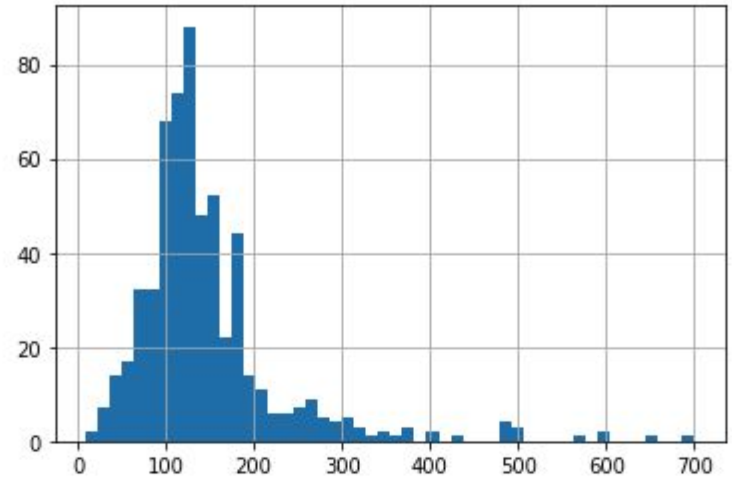
Data: Pre-Processing

Step 3: Look for trends and outliers among continuous variables with matplotlib library

Applicant Income



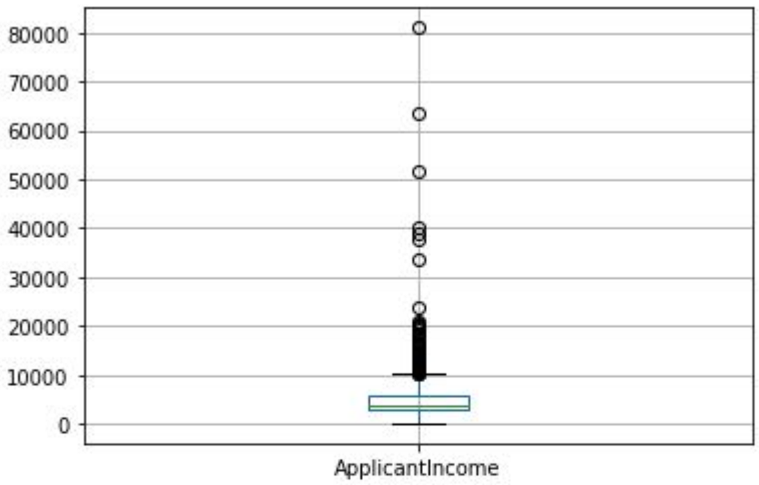
Loan Amount



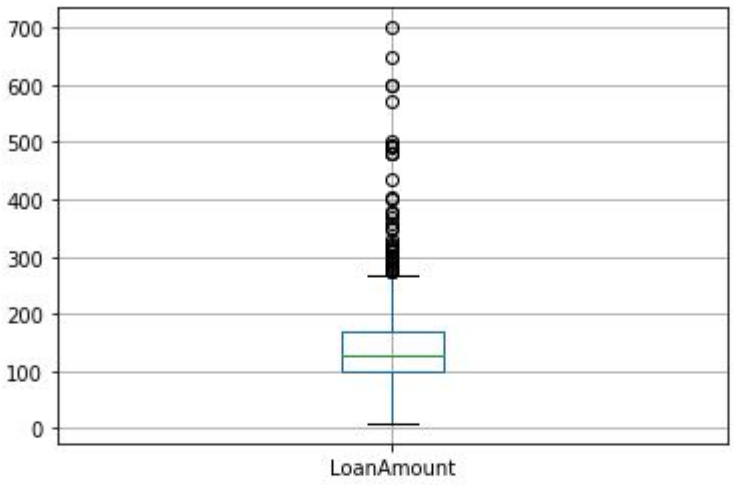
Data: Pre-Processing

Step 3: Look for trends and outliers among continuous variables with matplotlib library

Applicant Income



Loan Amount



Data: Pre-Processing

Step 4: Identify missing variables using pandas library

```
df.isnull().sum()  #output number of missing values in each column
```

Output:

```
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
```

Data: Pre-Processing

Step 5: Handle missing values

```
#imputing missing values with mean for continuous variables and mode
for categorical variables
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(),
inplace=True)
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0],
inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0],
inplace=True)
```

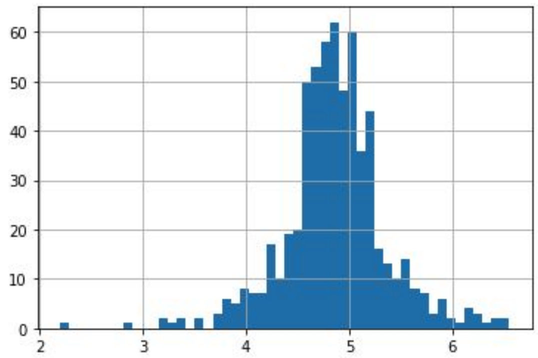
Data: Pre-Processing

Step 6: Handle outliers in continuous variables using numpy library

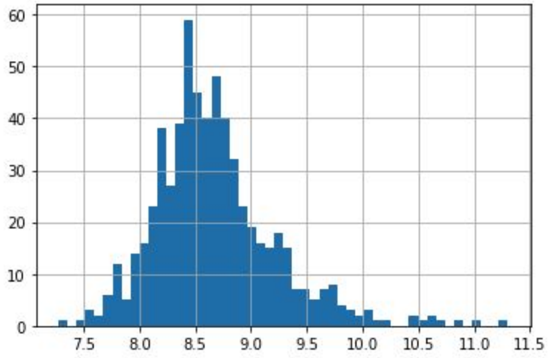
```
#create totalIncome column: sum of ApplicantIncome and CoapplicantIncome
df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']

#handle outliers by replacing continuous variables with log scale
df['TotalIncome_log'] = np.log(df['TotalIncome'])
df['LoanAmount_log'] = np.log(df['LoanAmount'])
```

Total Income (log)



Loan Amount (log)



Data: Pre-Processing

Step 7: Convert categorical variables into numerical values using sklearn library

```
# create array of categorical variables
cats =
['Credit_History','Dependents','Gender','Married','Education','Propert
y_Area','Self_Employed','Loan_Status']

#use label encoder to transform cats values to numerical values
for i in cats:
    le = LabelEncoder()
    df[i] = le.fit_transform(df[i].astype('str'))
df.dtypes
```

Method and Evaluation

Step 1: Create function for using model and outputting performance using sklearn library

```
def model(modelType, dataFrame, variable, outcome):
    modelType.fit(dataFrame[variable], dataFrame[outcome])
    predictions = modelType.predict(dataFrame[variable])
    accuracy = metrics.accuracy_score(predictions, dataFrame[outcome])
    print ("Accuracy : %s" % "{0:.3%}".format(accuracy))

    kf = KFold(5, True, 1)
    error = []
    for train, test in kf.split(dataFrame):
        train_variables = (dataFrame[variable].iloc[train,:])
        train_target = dataFrame[outcome].iloc[train]
        modelType.fit(train_variables, train_target)
        error.append(modelType.score(dataFrame[variable].iloc[test,:],
        dataFrame[outcome].iloc[test]))

    print ("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))
    modelType.fit(dataFrame[variable],dataFrame[outcome])
```

Method and Evaluation

Step 2: Combine test and train data into one dataset

```
df['Type']='Train'
dftest['Type']='Test'
dfcombined = pd.concat([df,dftest],axis=0, sort=True)

dfcombined.isnull().sum()
```

Step 3: Find missing values in combined dataset

Step 4: Impute missing values in combined dataset

Step 5: Use labelEncoder on categorical values in combined dataset

Method and Evaluation

Step 6: Use function with Logistic Regression and various predicting variables

- Model: `LogisticRegression()` from sklearn library
- Dataframe: `dfcombined` - combined dataset with train and test data
- Variables: `Credit_History, Education, LoanAmount`
- Outcome: `Loan_Status`

OUTPUT:

```
Accuracy : 80.945%
Cross-Validation Score : 80.944%
```

Room for Improvement

How to get more accurate results

- Test different metrics
- Test different hyper-parameters
- Test multiple models/algorithms
- Algorithm tuning
- More data