Logistic Regression of Credit Risk Assessment Using Credit Applications

Sydney Swofford | Final Project | CS7050

Dataset

The Kaggle Credit Data Set is an open-source data set that includes thousands of applicants for new lines of credit. Each applicant in the dataset contains the following information:

- SeriousDelinquencyIn2Years → Boolean target variable
- UtilizationofUnsecureCreditLines → Total balance on credit cards & loc/ # of credit lines
- Age → Applicant age in years
- NumberofPastDue30-59Days → # times past due between 30-59 days
- DebtRatio → Sum of debt/monthly gross income
- MonthlyIncome → Gross monthly income
- NumberofOpenCreditLines → Number of open loans
- NumberOfTime90DaysLate → # times 90 days or more late
- NumberOfRealEstate → # real estate loans
- NumberOf50-89DaysLate → # times late between 50-89 days
- NumberOfDependents → # dependents in family

Based on this information, the Boolean target variable of whether an applicant will be delinquent in two years will be predicted from the model.

Analysis

The Kaggle dataset was analyzed utilizing pandas and sklearn libraries in python, and the following steps were taken to create, fit, and test the model:

- Import Kaggle Give Me Credit Dataset as a Pandas DataFrame

```
# Training & Validation
train_file_path = "C:\\Users\\Sydney\\Documents\\GradSchool\\CS_7050 Data
train_data = pd.read_csv(train_file_path)

# Testing
test_file_path = 'C:\\Users\\Sydney\\Documents\\GradSchool\\CS_7050 Data_
test_data = pd.read_csv(test_file_path)
```

- Split the set into target variable and feature data x_train and y_train

```
# Split Feature & Target
X_train = train_data.drop('SeriousDlqin2yrs', axis=1) #features
y_train = train_data['SeriousDlqin2yrs'] #targetn variable
```

- Split the training data into training and validation sets

```
# Split training into training & validation
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=4
```

- Standardize the feature data using StandardScaler

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_split)
X_val_scaled = scaler.transform(X_val)
```

- Create Logistic Regression Model using sklearn

```
model = LogisticRegression(random_state=42)
```

- Fit Logistic Regression Model using the training feature data and target variable

```
model.fit(X_train_scaled, y_train_split)
```

- Use model.predict to make predictions

```
y_val_pred = model.predict(X_val_scaled)
```

- Evaluate predictions by comparing the actual values to the predicted values

```
accuracy_val = accuracy_score(y_val, y_val_pred)
```

- Use model to predict delinquency for entire future users and testing dataset

```
X_test = test_data.drop('SeriousDlqin2yrs', axis=1)# features only in testing set
X_test_scaled = scaler.transform(X_test)
y_test_pred = model.predict(X_test_scaled)
```

Results

After completing this project, it's obvious to me that the data set can be modeled relatively accurately using logistic regression. Logistic regression is historically good at estimating the relationship between dependent and independent variables, and I think my model demonstrates this. My model's confusion matrix is as follows:

Confusion Matrix

TrueNegative	FalsePositive
27973	71
FalseNegative	TruePositive
1881	75

Based on the above confusion matrix, the logistic regression model accurately reported 27,973 applicants as not being a delinquent threat in two years, and accurately reported 75 applicants as being a threat. The model inaccurately reported 1,881 applicants as not being a threat when they should have been flagged, while the model inaccurately reported 71 applicants as being a threat, when they should not have been flagged. Overall, the precision of the model when the user was not delinquent was 94%, however the precision when the user was delinquent was 51%, displaying that the model had a harder time deciding if a user was delinquent than if they weren't delinquent. The overall accuracy of the model was 93%, which is quite good regardless of the model struggling to flag delinquents. In my proposal I aimed to use the model to predict future delinquency of users and using a testing data set with approx. 100,000 entries, my model using decides whether given the 10 categories of user information [excluding target variable] if the account will be delinquent or not. Inputting my own information into the testing set, I can see the likelihood the model will predict if I will be delinquent or not! The model predicted I would not be delinquent, and out of my own curiosity, since my model is roughly 94% accurate, I went ahead and applied to a credit card application to see if I would get approved, as my model has assured me that I will. As with most credit card applications these days, the results take about as long as my python script to run, and I was in fact approved, further validating the results and predictive ability of my model!