Loan Grading Web App

Estefany Amado
Jenny Bui
Tania Guevara
Josh Pardo
Meggie Pires-Fernandes

Proposal

Overview:

We'll be using existing credit loan data from Lending Club to train a model to predict if a loan candidate will be likely to repay the loan.

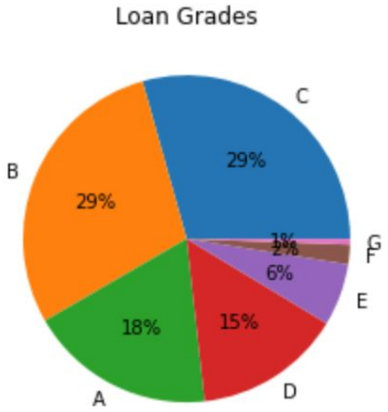
Our target value will be loan grade, which is a classification system that assigns a quality score to a loan application to identify a risk of default.

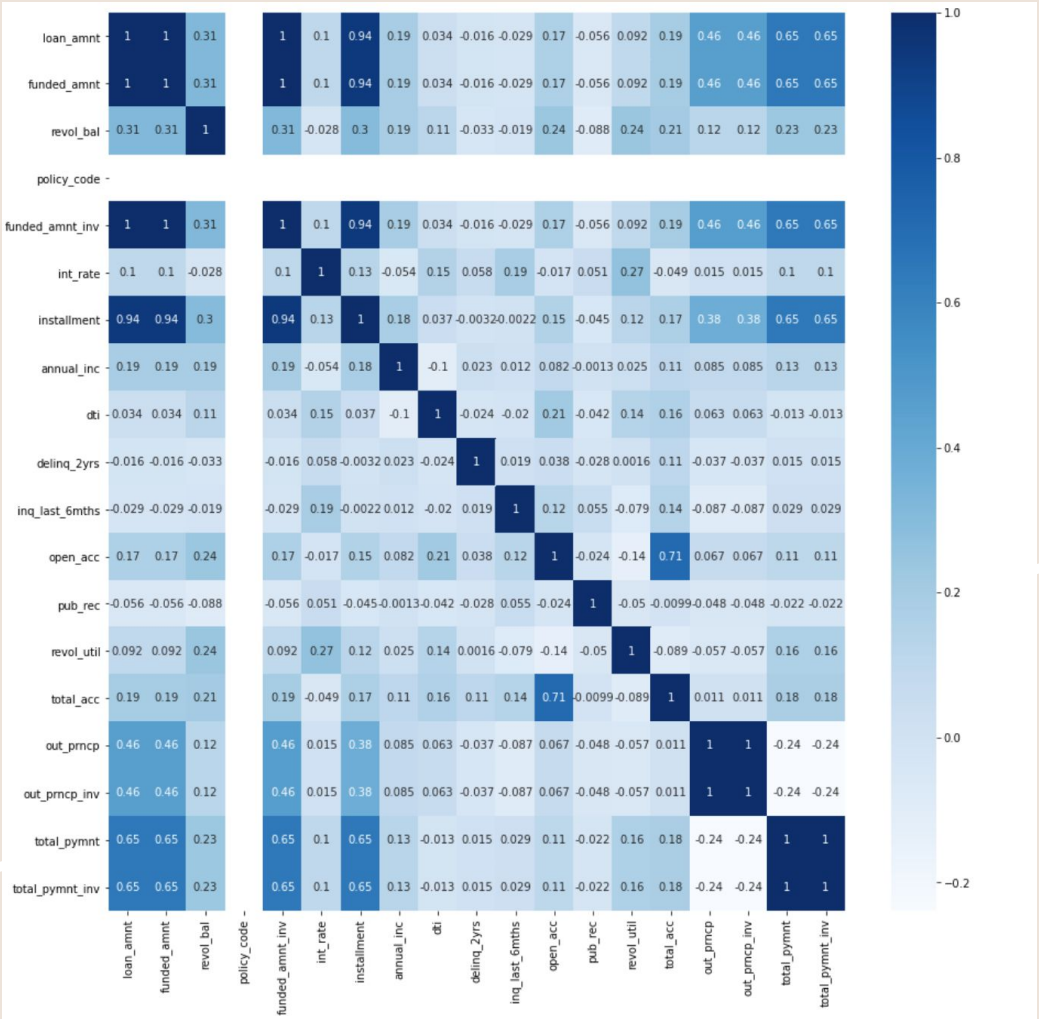
Project Outline:

- Use Lending Club Loan Data to create decision tree predictive model
- Create, implement, and optimize a predictive model using:
  - Python Pandas
  - Sklearn
  - Python Matplotlib
  - SQL
- Utilize HTML/CSS/Bootstrap to create a webpage to display visualizations and loan prediction app
- Deploy app to Heroku

Building, implementing, and optimizing our model

- Preprocessing:
  - Data Cleaning
    - Dropping null values
  - Data Transformation
    - Modifying the data so that the loan grades are numeric
  - Data/dimension reduction
    - Creating correlation map to determine key factors





Based on the correlation map, the factors that are most correlated with the grade are:

- Interest rate
- Total payment
  - Payments received to date for total amount funded
- Revol_util
  - The amount of credit the borrower is using relative to all available revolving credit
- Inq last 6 months
  - # of inquiries in the past 6 months (excluding auto and mortgage inquiries)
- DTI(Debt to Income Ratio)
  - Ratio calculated using the borrower's total monthly debt payments on the total debt obligation divided by the borrower's self reported monthly income

- Compile, train, and evaluate a machine learning classifier
    - Train_features: train data extracted features
    - y_train: train data labels
    - Test_features: train data extracted features
    - Y_test: train data labels
    - Return:
        - Results(dictionary): A dictionary of a classification report
    - Models:
        - Decision Tree Classifer
        - Grid Search CV
- Save and export our model as a serialized object pickle file

Creating our front-end web page and loan prediction form using HTML/CSS

- Utilized HTML/CSS/Bootstrap to create a web page to hold our visualizations developed when building, implementing, and optimizing our loan grade predicting model
- One of our tabs features a test form that will generate an individual's predicted grade determining the borrower's ability to repay their loan

Deploying the web app using Flask

- Utilized Flask to build our web application and run it locally

```python
import pandas as pd
import numpy as np
# import sqlalchemy
import csv
import time
import pickle
# from config import password
from decimal import Decimal
from flask import Flask, render_template, redirect
from flask import Flask, redirect, url_for, request
# from flask_sqlalchemy import SQLAlchemy
from flask import Response
from flask import request
# from sqlalchemy.ext.automap import automap_base
# from sqlalchemy.orm import Session, session
# from sqlalchemy import create_engine, func
# from sqlalchemy_utils import database_exists, create_database
from flask import Flask, jsonify

# app = Flask(__name__)
# app.config['SQLALCHEMY_DATABASE_URI'] = f'postgresql://postgres:{password}@localhost:5432/project4'
# sql = SQLAlchemy(app)

# Creating Engine
# engine = create_engine(f"postgresql://postgres:{password}@localhost:5432/project4")
# if not database_exists(engine.url):
#     create_database(engine.url)

# # reflecting database
# Base = automap_base()

# # reflecting tables
# Base.prepare(engine, reflect=True)

# Load ML model
model = pickle.load(open('decision_tree_classifier.pkl', 'rb'))

# # Load initial data to sql
# data = pd.read_csv("clean_loan.csv")
# data["GRADE"].replace({"A":0, "B":1, "C":2, "D":3, "E":4, "F":5, "G":6 },inplace=True)
# try:
#     data.to_sql(name='initial_loan_data', con=engine, if_exists='fail', index=False)
# except ValueError:
#     pass

# Create application
app = Flask(__name__)

# Bind home function to URL
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/WebVisualizations/index.html')
def form():
    return render_template('WebVisualizations/index.html')

# Bind predict function to URL
@app.route('/predict', methods = ['POST'])
def predict():
```

Limitations:

- Because Lending Club no longer operates as a peer to peer lender as of December 31st, 2020, the data consists of past data from 2007-2020.
- Our data source was very large, so it was bit of a challenge downloading it to our Git Repo. We had to use the extension Github LFS that replaced the large file with text pointers inside Git.
- Also, the enormous CSV file used for our model caused Jupyter Notebook to run a bit slower, thus our model(s) took awhile to run/generate.

Final thoughts and conclusions:

- Ways our loan grading web app can be furthered:
  - Create a button where a CSV file can be uploaded into the model/app and generate predictions based on the inputted file
    - Where the model can keep learning
    - Input other type of loan data that isn't only from Lending Club
  - Can be expanded to cover not only home loans, but other type of loans like auto
  - Modify the model to not only predict and generate loan grade, but as well as interest rate

Thank you.