CSC461 Final Project Presentation

Brian Materne

Project Goals

The main goal of my project is split into several steps, all centered around finding and using a completed Kaggle competition. Once I found one, the plan was to:

- Analyze the data on my own terms
- Gather completed solutions for the top three submissions
- Analyze these completed solutions
- Extract specific details that could improve lower ranked solutions
- Apply extracted details to lower ranked solutions to improve them

Finding my Dataset

Finding a dataset was relatively simple, I was able to search through completed competitions, and I found one from Kaggle's **playground series**. This is a series of no-reward competitions, with large amounts of entries and a very low barrier of entry. This means I'd have people implementing very good analyzable solutions, but also a large amount of middling solutions that are able to be improved.

The specific competition I found was on a **loan approval prediction** dataset. It just took place this last October, so everything is current and very recent. There were around 4,000 different teams with almost 30,000 submissions, so this competition was perfect for what I wanted to do.

Data Analysis

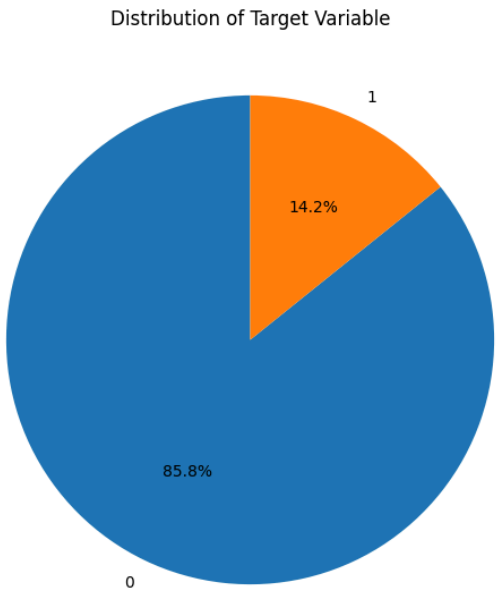
The data being used is on **loan approval prediction**, where the features have information about a loan (both about the loan itself and about the person), and the target is whether or not the loan was predicted.

Shape/Balance

```
# Gather / print data shape
print(train.shape)
print(test.shape)
```

```
(58645, 12)
(39098, 11)
```

Both sets have **11 feature columns.** The training data has **58,645 rows**, and the testing data has **39,098 rows**. Since this is a competition, the target column does not exist within the testing set. The only way to verify answers is through cross validation within the training set or with a small section of the testing set which served as a “public leaderboard” until the competition ended.

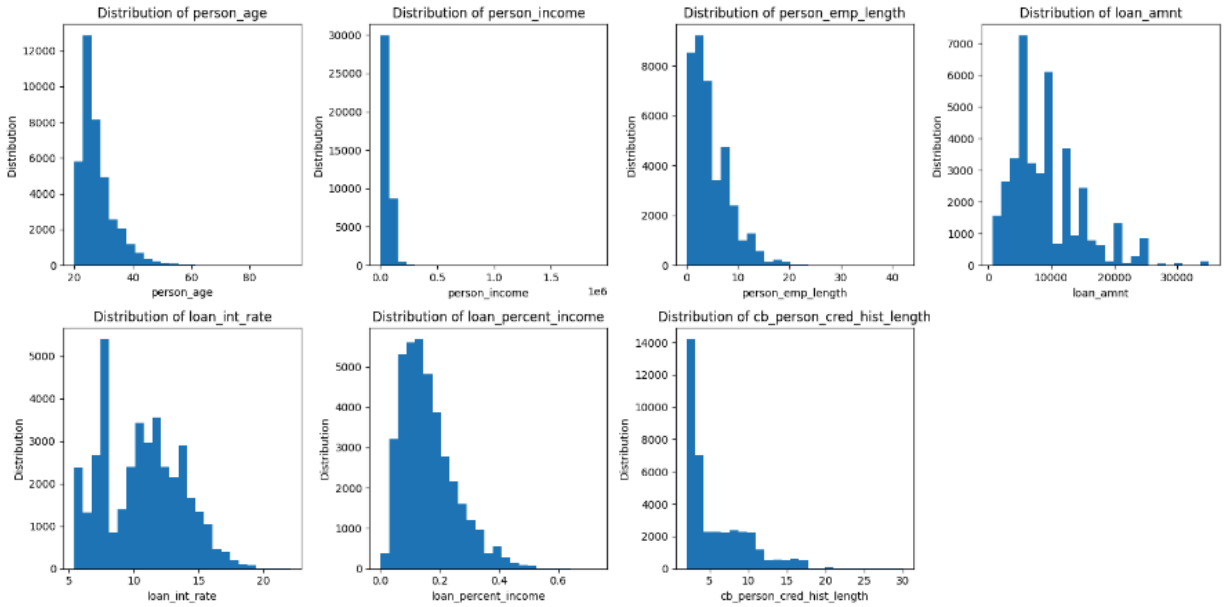


The dataset is not balanced, and has a much higher distribution towards a loan being denied. This makes sense for the given question, and makes the dataset a good amount more unconventional.

Feature Analysis

Because the dataset is made up of both numerical and categorical features, I decided to analyze each set separately. Each of these analyses are done through a combination of both the testing and training features.

```
# Extract all numerical features from train and test
train_numerical_data = train.select_dtypes(include=['number']).drop(columns=['loan_status'], inplace=True)
test_numerical_data = test.select_dtypes(include=['number'])
numerical_data = pd.concat([train_numerical_data,test_numerical_data],axis=0)
```

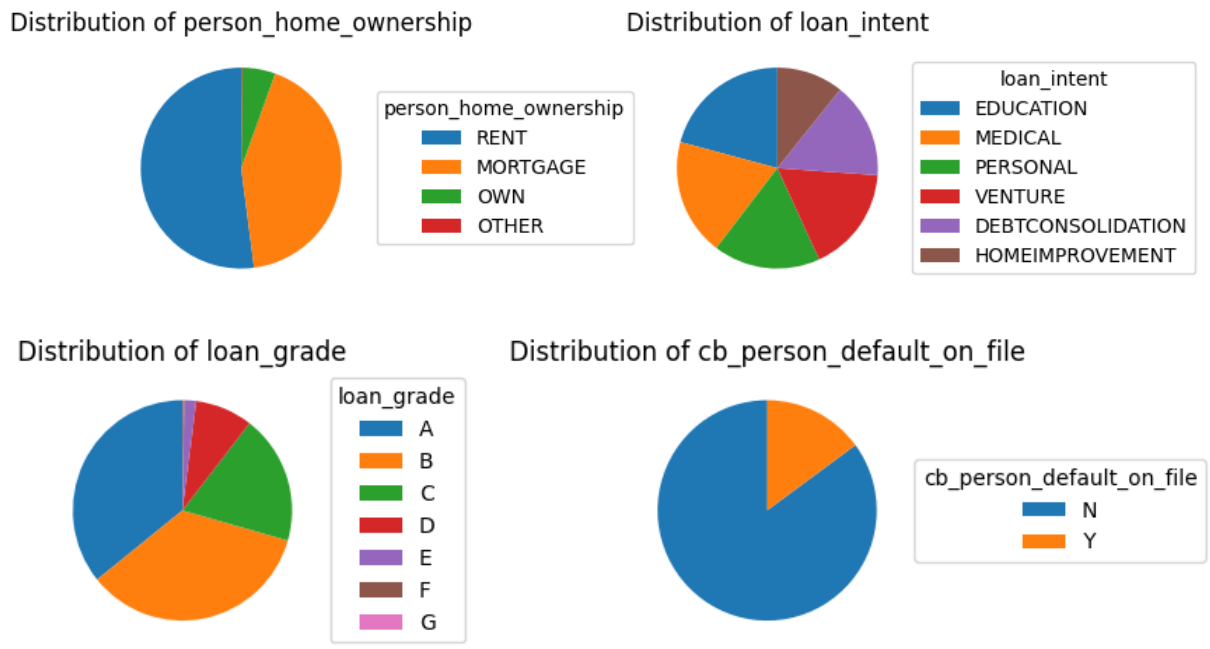


Apart from some anomalies, we can see this data is mostly unimodal. The exceptions are loan interest rate being multimodal and some minor fluctuations in loan amount, likely caused by loan amounts gravitating to “friendly” numbers.
We can also see from this what types of features are in this dataset. We have features entirely about a person, like “person_age”, and features entirely about the loan, like “loan_int_rate”. The set uses these personal and loan-based details in order to get a full picture of the loan being asked.

On the other hand, the categorical features were analyzed in a similar way.

```
# Extract all categorical features from train and test
train_categorical_data = train.select_dtypes(include=['object', 'category'])
test_categorical_data = test.select_dtypes(include=['object', 'category'])

categorical_data = pd.concat([train_categorical_data, test_categorical_data], axis=0)
```



We can first see here that the types of data are consistent with the types of data from the numerical columns. We can also learn some things about these points themselves.

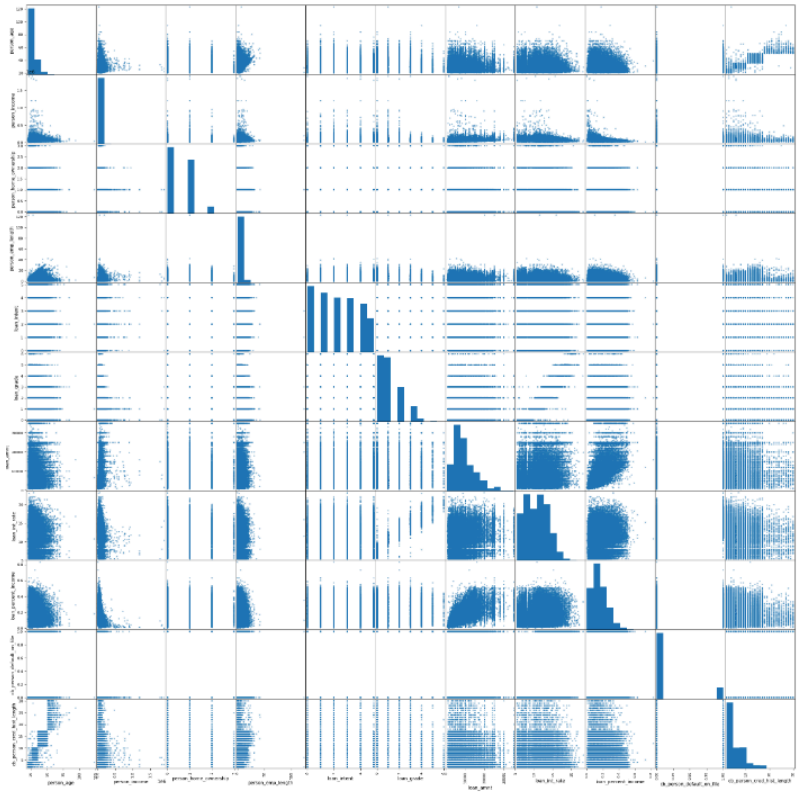
The only balanced column among all four categorical columns is loan intent, with a near balanced split. Every other column is heavily in favor of one or two different outcomes, speaking to the unbalanced nature of this dataset.

We can also see an “OTHER” option in the home ownership column. This isn’t indicative of multiple possible answers combined together, “OTHER” is just one of the four possible types.

Feature Comparison

After individual analysis was complete, the next step was to make a scatter matrix between features. This first one does not include the target column and consists of both the training and testing data.

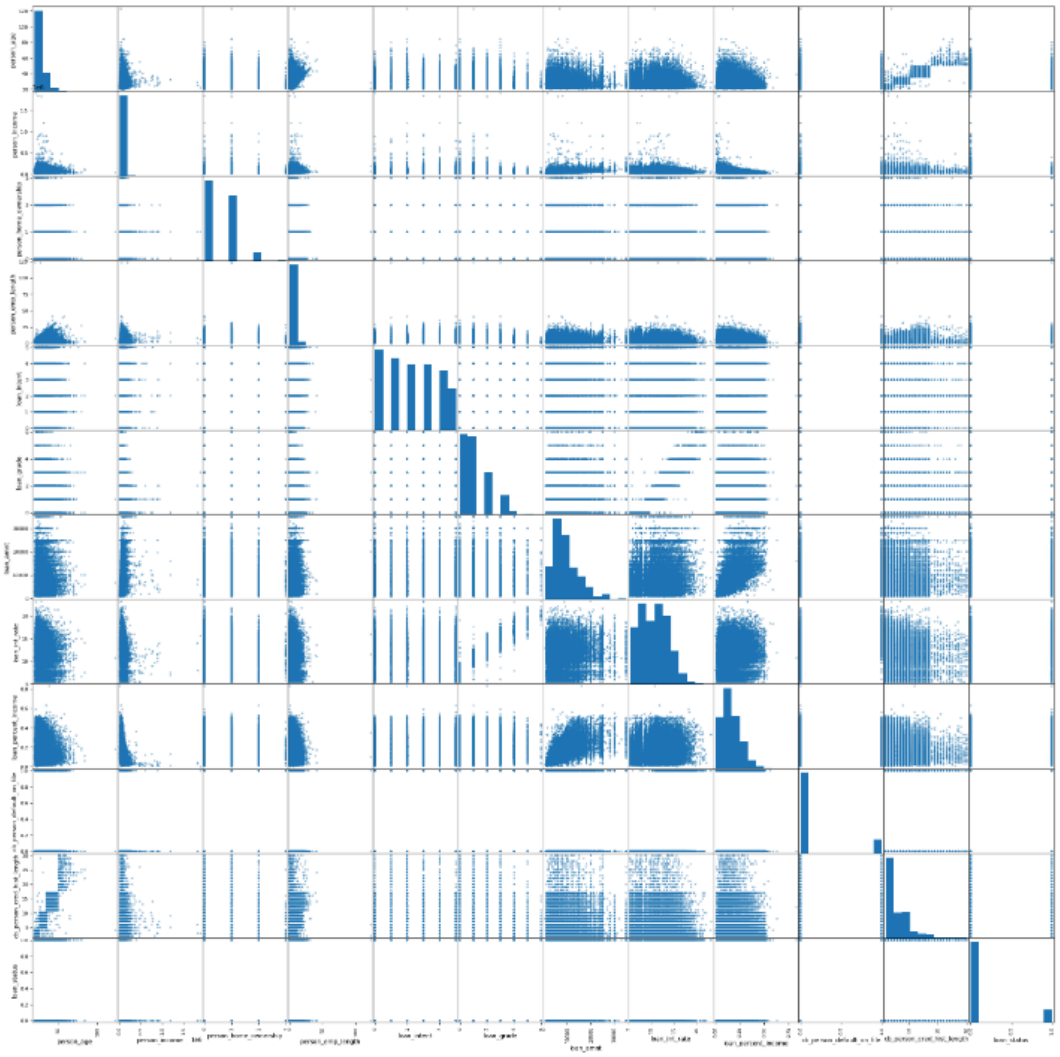
```
# Concatenate the train and test sets, getting rid of the target
features = pd.concat([train.drop(columns=['loan_status']),test],axis=0)
```



As we can tell from this, the plot is very messy. Barely any features actually trend alongside other features, it all feels random or self-contained for the most part.
We can see a couple of diagonal distributions indicating trends, but it's nothing that isn't expected. (Credit history length scaling with age)
The goal for this experiment was to see if any features scaled with other non-obvious features, preferably bridging the gap from person to loan information, but no evident connection was able to be drawn.

After this, I created a second scatter plot with the addition of the target. This was just to gather each feature's trends against the target.

Since this requires the target to be present, only the training data is used.



While it's hard to see with the graph this zoomed out, the target column is the bottom row of the matrix and the rightmost column of the matrix.

Not much can be gathered, it seems that the features are mostly independent of the target. There's a slight lean towards higher approval chances with higher age and income, but nothing besides that is discernible from the matrix.

Top Submissions

The next step was to gather the top submissions from the Kaggle competition results. Since there was no public submission for the third place solution, I'll be analyzing the first, second, and fourth place solutions.

Each solution is properly credited on my created GitHub, and the solutions and full analyses are also present.

Goal

When analyzing each solution, I'll have two goals in mind.

The first goal is to understand the systems at play. In order to use any methods from a solution, I want to know exactly what is going on. This is especially important when it comes to the methods that set it apart from other solutions.

The second goal is to isolate extractable features that can be applied to other solutions. This requires an understanding of the solution at hand, an understanding of the dataset, and an understanding of which specific parts are applicable to other solutions.

Ideally, the parts that make a solution fit a top three slot are extractable and applicable, as that makes the entire process fully smooth. If that isn't the case, minor stepping stones of the given solution are able to be extracted and combined to bolster another solution.

First Place Solution

The first place solution, called CatBoost All the Way, used CatBoost along with a variety of other techniques to achieve its score.

Cross Validation

The first realization in this solution is how cross-validation metrics should be used. Since this is a competition, before the deadline is reached, the entirety of the testing set doesn't have its target available for use. All people have to work with is a small public portion and their cross-validation scores.

This solution realized that, instead of using the public test accuracy, it's much more effective to set up a good cross-validation scheme and rely on cross-validation exclusively to gauge different models.

They used a five-fold validation set with two repeats, creating a total of ten splits.

Model Selection

The models that were chosen were (as the name suggests), CatBoost, as well as LightGBM, XGBoost, and a standard neural network. Each of these was trained and averaged through different iterations of Optuna. CatBoost was found to have the best performance in the first run, so a new CatBoost model was trained using the results of each of the four previous models, eventually, a CatBoost model trained on the previous CatBoost model was able to create the optimal score.

LightGBM / XGBoost

While they aren't important for why this solution works, LightGBM and XGBoost are helpful to learn why CatBoost works so well.

We talked about AdaBoost in class, and these are both similar models. They're both gradient boosting machine learning frameworks that use decision tree forests.

One important feature to know is that neither of these have inherent categorical feature support, as they only work with numeric features. If a categorical feature is used, it gets converted into numeric values.

CatBoost

The difference between CatBoost and these other boosting algorithms is that it's designed around the use of categorical features, and supports them natively. This makes it support the multiple categorical features in the data. On top of that, this solution does something extra by counting multiple of the numeric features as categorical features. This allows for a different interpretation of the data, which ends up being advantageous, and it allows a better use of CatBoost.

Optuna

Optuna is a program that allows you to adjust the hyperparameters of a model being trained. This saves you the process of manually tuning hyperparameters, and provides a more streamlined approach to stochastic search. In this solution, Optuna was used on each model over multiple passes, and an average for each parameter was taken.

Second Place Solution

The second place solution was really simple, consisting entirely of a large ensemble of models with a hillclimber model at the end. As we saw in the first place solution, CatBoost was also the most prominent model in this solution. While not as important to how this solution performed, the other models used were:

- LightGBM
- Dart
- XGBoost
- ExtraTrees
- Random Forest
- KNN
- MLP
- Goss

Hillclimber

In large ensembles like this, there's a chance that models are present that are bringing down the overall predictions. There are multiple models to mitigate this, and the one used in this solution is **hillclimber**.
It starts with the highest performing model, and sequentially adjusts parameters and adds new models. If any negative progress is made, hillclimber negates the addition. This means only the positively impacting models stay, and anything that isn't beneficial isn't included. Because of this, any number of models can be added with no detriment.

Fourth Place Solution

Since the third place solution doesn't have a public notebook, the last one being analyzed is the fourth place solution. Most of this notebook is pure data analysis, and the actual solution is very simple. Four models are created, nothing unique from the previous solutions, and an ensemble of them by itself is the final solution. However, one unique practice is employed.

Data Manipulation

One unique method this solution employs is its use of data manipulation. Throughout the feature analysis, a new feature was added as a combination of two previous features. Loan amount, loan person income, and person income all get combined into "loan to income" as one extra feature.

Takeaways

After looking at each of the top three solutions, we have a list of extractable components that can be used in other solutions. These are:

- Choice of Models
    - CatBoost as the most effective
    - LightGBM and XGBoost behind
- Data Transformation
    - Categorical transforms
    - "Loan to income" column
- Optuna
- Hillclimbing

With all of these, we have a repertoire we can apply to other solutions.

Low Model Boosting

Unfortunately, no solutions outside of the top minority had linked solutions, so I wasn't able to apply these techniques to any real solutions. However, we have enough information to understand the types of solutions that can be boosted with what we've gathered.

- If a solution only uses one model, we can boost it by adding more models and creating an ensemble. The most helpful models to be added are CatBoost, as well as the other gradient boosting frameworks.
- In any solution with an ensemble, it can be worth checking if Hillclimbing increases the general accuracy.
- For any model, we can run it through Optuna to see if any hyperparameters can be optimized from their current values
- In any case, we can try feature transforms in order to check their relative accuracy. This can be both creating a new column or converting numeric columns to categorical ones.

Summary

Overall, when given any subpar solution to this problem, we have a list of edits and additions that will definitely improve the solution in one way. While I haven't been able to concretely apply these tactics, we can understand conceptually and based on the results of the other submissions to what extent these additions will work.
