jpetoskey / Loan_App_Amended Public

0 stars 0 forks

Star Unwatch

Code Issues Pull requests Actions Projects Wiki Security Insights

main

jpetoskey ... 10 seconds ago

View code

Loan_App_Amended

Overview

Use Kaggle for Data Source: Kaggle - Credit Analysis using EDA on Loan Applications.

Predict approval of loans with a high True Positive Rate, in order to build a filter for loan applications, so that applications passed on to people have a higher rate of being approved, thereby lessening the human-hours needed to filter loan applications.

This model will save banks time and money by filtering loan applications. And, with future iterations, can be tuned to increase the likelihood of repayment.

Business Problem

- Banks want a simple and efficient way to filter cash and revolving loan applications:

- Prior loan applications allow models to make strong inferences on which loans will be approved in the future, thereby saving banks time and money. In addition, the model can be tuned to have banks make more money, with the addition of features such as repayment.

Data

The data for this project comes from Kaggle. It is a fairly large dataset comprised of 122 Features and 307,511 loan applications. Some of the important features include, date of birth, gender, education, amount of annuity, in addition to features that aren't explained, such as EXT_SOURCE 1, 2, and 3.

Data Source: Kaggle - Credit Analysis using EDA on Loan Applications.

Some data cleaning and manipulation was required. There were missing values in 19 numerical features, which were replaced with the mean from each feature. In addition, there were 242 missing values in one categorical feature 'NAME_TYPE_SUITE', which were replaced with the mode -'Unaccompanied'.

Due to a class imbalance in the target variable - Loan Approval - which contained 8% Loan Approvals, the majority class -Refusal- was undersampled and the minority class -Approval- was oversampled. Undersampling was completed by choosing a random 40,000 values from the '0' or refusal class and oversampling was completed using SMOTE to increase the minority class from 24,825 to 40,000 values.

In the train, test, split, 40% of the data was saved for testing, due to issues with consistent results in the test set.

Methods

This project was my first exercise in classification modeling, so I chose to practice many forms of classification modeling before settling on my favorite modeling type for this data set. I ran multiple Logistic Regression, KNN, bagged trees, boosted trees, XG Boost, Random Forest, Gradient Boost, and Gaussian Naive Bayes models before landing on Random Forest and Gradient Boost as the top modelling types. I would have experimented more with KNN, but the computational power required for this data set was larger than practical for my machine. In that respect and disappointingly, there were a few modeling techniques, such as Recursive Feature Selection, that were too computationally expensive to run, as well.

However, I was able to perform feature selection and recommend feature importance using sklearn's feature_importances_. The feature imporance object was computationally inexpensive and allowed me to recommend the top features in the random forest model.

The final model was produced with sklearn's ensemble voting classifier and was comprised of the random forest and gradient boost models with the highest true positive rates, in addition to the random forest model with the highest true negative rate. The random forest with a high true positive rate was given a higher weight of 2, compared to a weight of 1, for the other two models, to optimize for a higher true positive rate in the final model.

Results

Final Model True Positive Rate: 84%

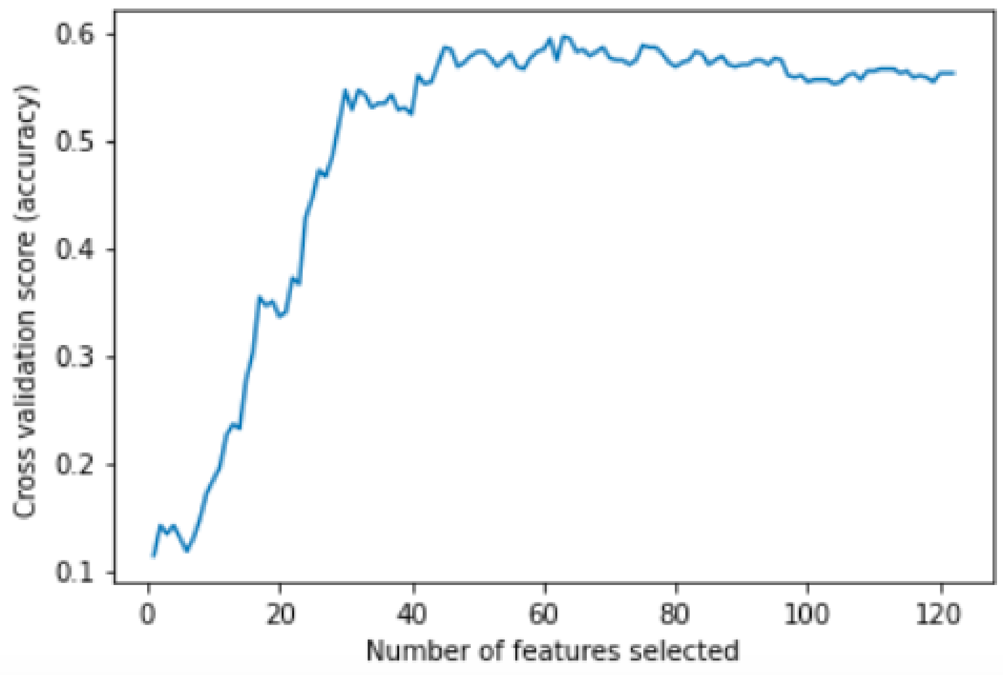
- 84% of the applications that were approved, were flagged for approval by the model.

Final Model True Negative Rate: 40% *31% of applications were filtered out by the Final Model, and only 14% of approved applications would have been filtered out with that data.

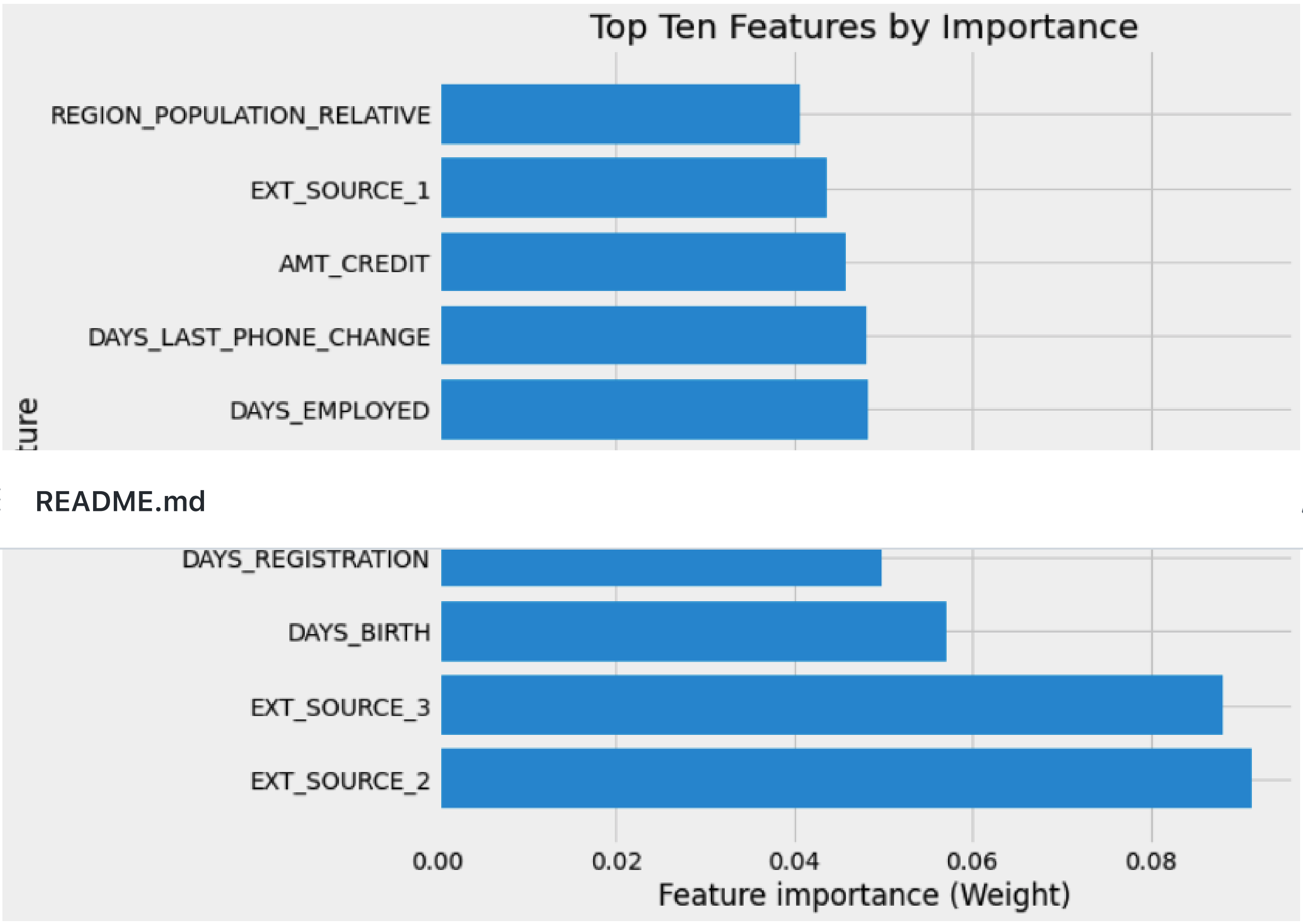
The Final Model did the best job of optimizing for a relatively high True Positive Rate, while correctly filtering out as many applications as possible.

Visual 1

```
Optimal number of features : 63
```

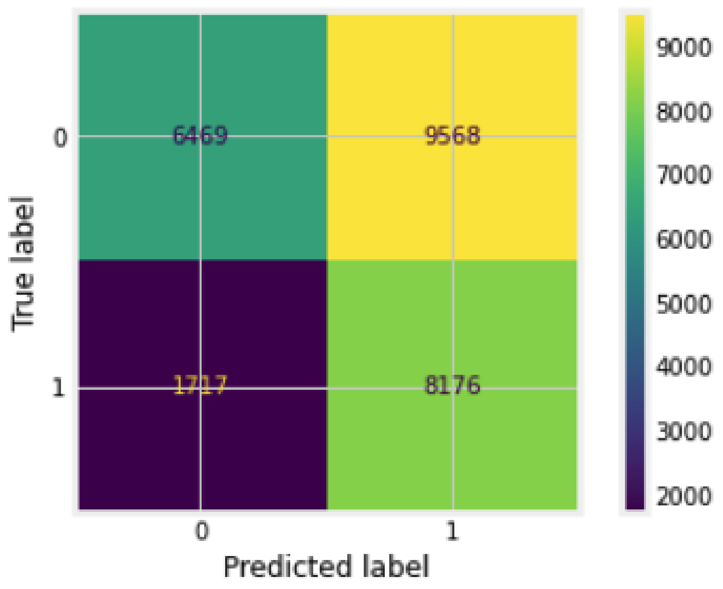


Visual 2



README.md

Visual



Conclusion

Banks can now filter out 31% of cash and revolving loan applications and be assured that 84% of approved applications will still be in the application pool to be reviewed by loan officers.

By filtering out 31% of applications, banks will be able to save time and money, and simplify the process for loan approval by their loan officers.

Loan officers will now see an application pool where 46% of applications should be approved, rather than an initial pool where 8% of applications were approved.

Further Studies Recommended

Gather data on more relevant features

- Credit Score
- Cash and Invested Savings
- Retirement Savings
- Debt to Income Ratio

Test model on other samples of market data

- Iterate and improve model for these samples or market segments

Improve pipeline for loans by developing digital questionnaire

- Attain data inexpensively – directly from consumer
- Tailor questions to garner data from most relevant features

For More Information

Please review my full analysis in Jupyter Notebook or my presentation.

For any additional questions, please contact **Jim Petoskey - Jim.Petoskey.146@gmail.com**

Repository Structure

```
├── README.md                           <- The top-level README for
reviewers of this project
├── Loan_App_Condensed.ipynb            <- Narrative documentation of
analysis in Jupyter notebook
├── Loan_Application.ipynb            <- Extended First Draft, Narrative
documentation of analysis in Jupyter notebook
├── Loan_App_Presentation.pdf           <- PDF version of project
presentation
├── data                                <- Not present, as the file was
over 100 MB
└── images                              <- Both sourced externally and
generated from code
```

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- ● **Jupyter Notebook** 100.0%