Business Problem

For this project, I chose to predict the outcome of crowdfunding (specifically Kickstarter) campaigns. Despite the potential benefits of a successful crowdfunding campaign, many projects fail to obtain the funding they seek, and entrepreneurs would benefit from a tool to predict ahead of time whether or not their planned campaign was likely to reach its goal. Kickstarter funding is all-or-nothing – if the campaign does not reach its funding goal, the creator receives no money – making classification models an appropriate choice for this task.

Data Set and Feature Engineering

Data used for this project comes from the 2025-06-12 release of web-scraped Kickstarter data at https://webrobots.io/kickstarter-datasets/ (which was the most recent release at the time I began the project). This dataset included a good deal of relevant information, but was poorly formatted – rather than each value being in its own column, some columns of the downloaded CSV files contained JSON dictionaries with a number of related values. Some of these (such as usernames) contained characters like quotation marks and commas which caused JSON parsing to fail, making it difficult to extract the relevant information.

The features I extracted and included in my models were:

- **Discrete features (one-hot encoded)**: Project category, subcategory, creator country, month launched, day of week launched (parsed from launch date), presence/absence of video, prelaunch activation, staff pick
- **Numeric features**: Funding goal (converted to US dollars if given in non-US currency; the dataset included exchange rates at campaign launch time), campaign length in days (deadline - launch date), launch time (hours from noon), blurb length (in characters), campaigns previously backed by creator, campaigns previously launched by creator (calculated by ordering campaigns by launch date and taking the cumulative count for each creator ID).

Because the full dataset was very large (over 100,000 rows), model fitting using all data was unfeasibly slow on my computer. To keep computational resources required more reasonable, I filtered the dataset to include only campaigns launched in 2024 or later. I also removed a few campaigns with funding goals over $100,000 as outliers likely to distort model predictions. The final dataset included 33,385 campaigns, around 2/3 of which reached their funding goal.

Model Fitting

After splitting the data into train and test sets (70% train, 30% test, stratified by target), one-hot encoding discrete features, and scaling numeric features using a robust scaler since the distribution of the funding goal was non-normal, classification models were assessed on the training set. For the K Nearest Neighbors algorithm, k-values from 1 to 30 (with a step size of 4) were tested to identify an optimal value. The highest 5-fold cross-validation accuracy was obtained with k=29. It appeared possible that validation accuracy would continue to increase for higher values of k, but the increase as k approached this value was very slow (see figure in

Appendix), suggesting that the increased resources required would not be worth the incremental improvement in accuracy. Likewise, different numbers of estimators were tested for the Random Forest algorithm; differences between results were similar for 200, 500, and 1000 estimators. KNN with k=29 and Random Forest with 500 estimators were selected for cross-validation along with the logistic regression, Naive Bayes, and decision tree models, as well as support vector machine models with RBF, sigmoid, linear, and polynomial kernels. The 5-fold cross-validation results on the training set were highest for the Random Forest model (average accuracy = 0.8331) and lowest for the SVM models with sigmoid and polynomial kernels (average accuracy = 0.7128 for both). Since around 66% of the training data comes from successful campaigns, a model that always predicted success would be accurate 66% of the time, meaning 71% accuracy is very poor!

On the testing data, the Random Forest model again had the highest accuracy (0.83), followed by SVM with RBF kernel (0.81). However, the Random Forest model had a considerably higher false-positive rate than SVM (which had fewer false positives, but more false negatives). In this business context, false positives are more risky than false negatives, since a user could invest time and money into their Kickstarter campaign if given a prediction that it would succeed, which they would then lose if the campaign in fact failed. Choosing not to launch a Kickstarter campaign is less risky (and the user will never know if the campaign would have succeeded, so cannot blame the model for providing a false prediction). For this reason, I decided to implement the SVM model, rather than random forest, for my Streamlit application.

The K-Means model was fitted separately after testing the supervised learning models. The elbow method indicated that the optimal number of clusters was 4, rather than the 2 possible outcomes of the campaign. Examining the 4-cluster solution, three clusters contained mainly successful campaigns and one contained mainly negative campaigns. However, the accuracy that would be obtained by considering the first three clusters to be classified as “success” predictions and the last as “failure” would be 0.743, worse than most of the supervised models tested. Forcing a two-cluster solution would result in even worse classification accuracy, with the large majority of campaigns placed in the first cluster regardless of success.

Limitations

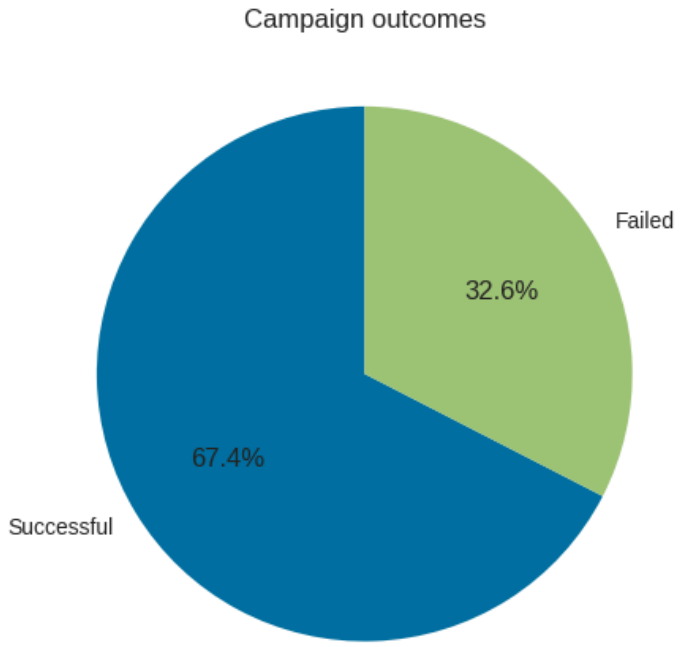
This data set does not include full project descriptions or reward tiers offered, which play a significant role in audiences’ willingness to back a Kickstarter project. I attempted to scrape Kickstarter myself to obtain this data, but was prevented from doing so by anti-bot measures. However, including this data (applying natural language processing to extract features from project descriptions) would almost certainly result in higher classification accuracy.

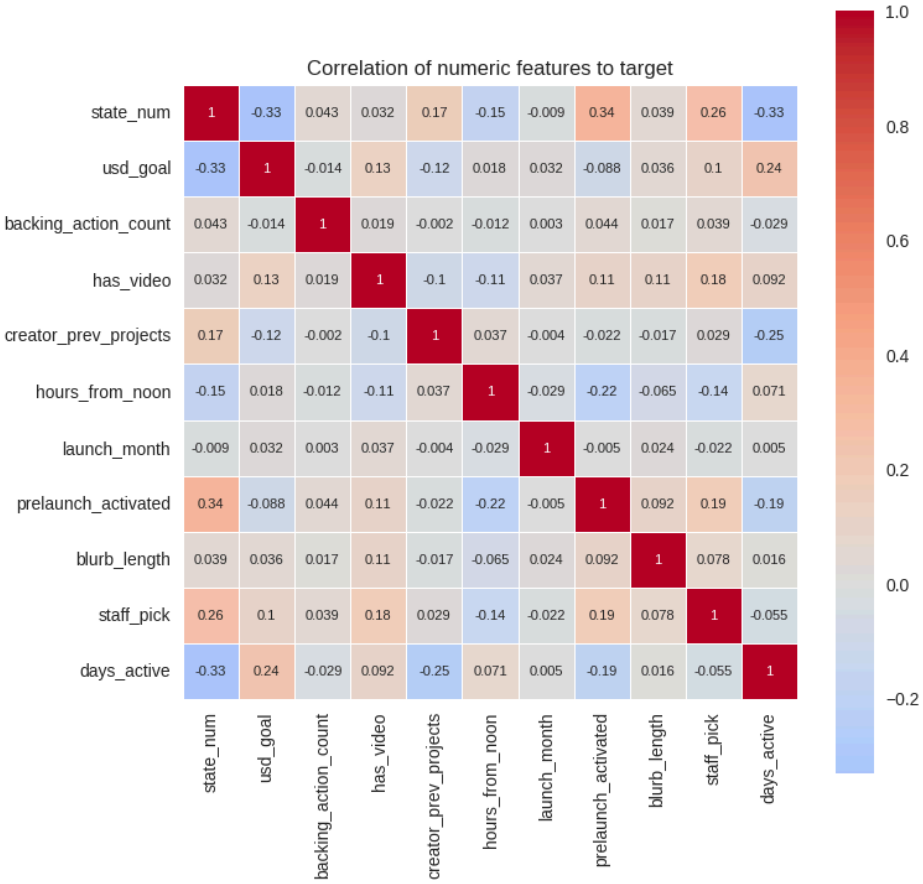
The chosen model (SVM with RBF kernel) is not highly interpretable – using the kernel trick improves model performance, but obfuscates the connection between specific features and predicted outcomes. It may not be intuitive to users which aspects of their planned Kickstarter campaign would be best to adjust to increase their predicted chance of success.

Appendix

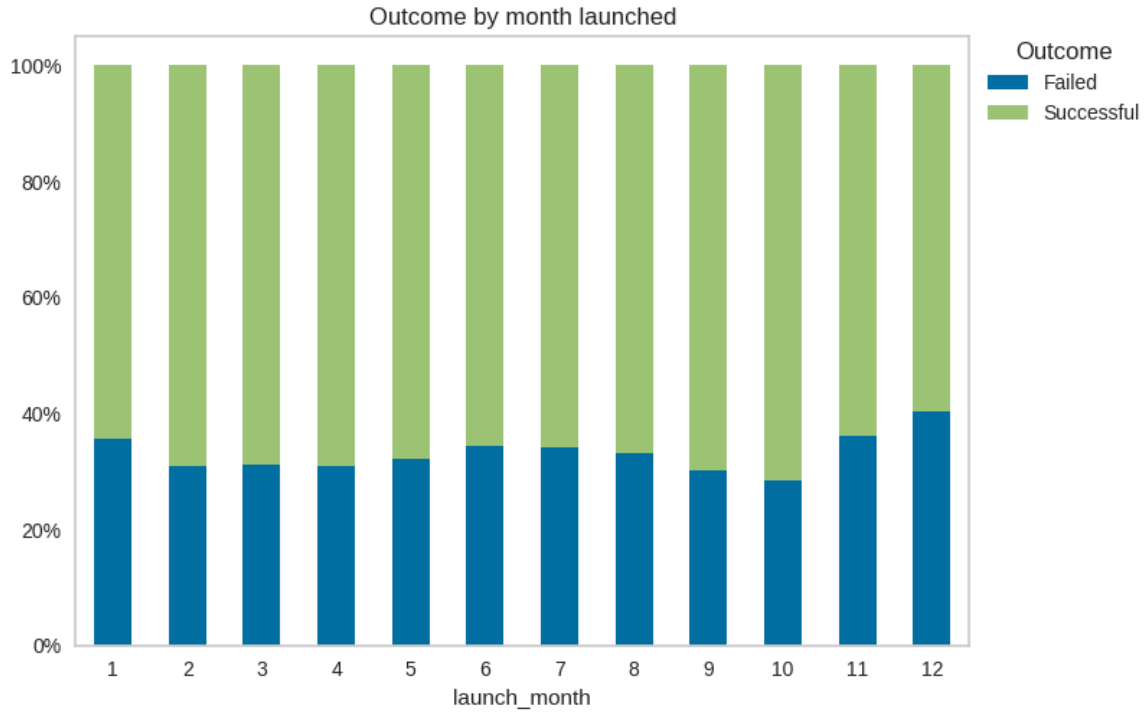
Data Visualization – selected feature and target distributions

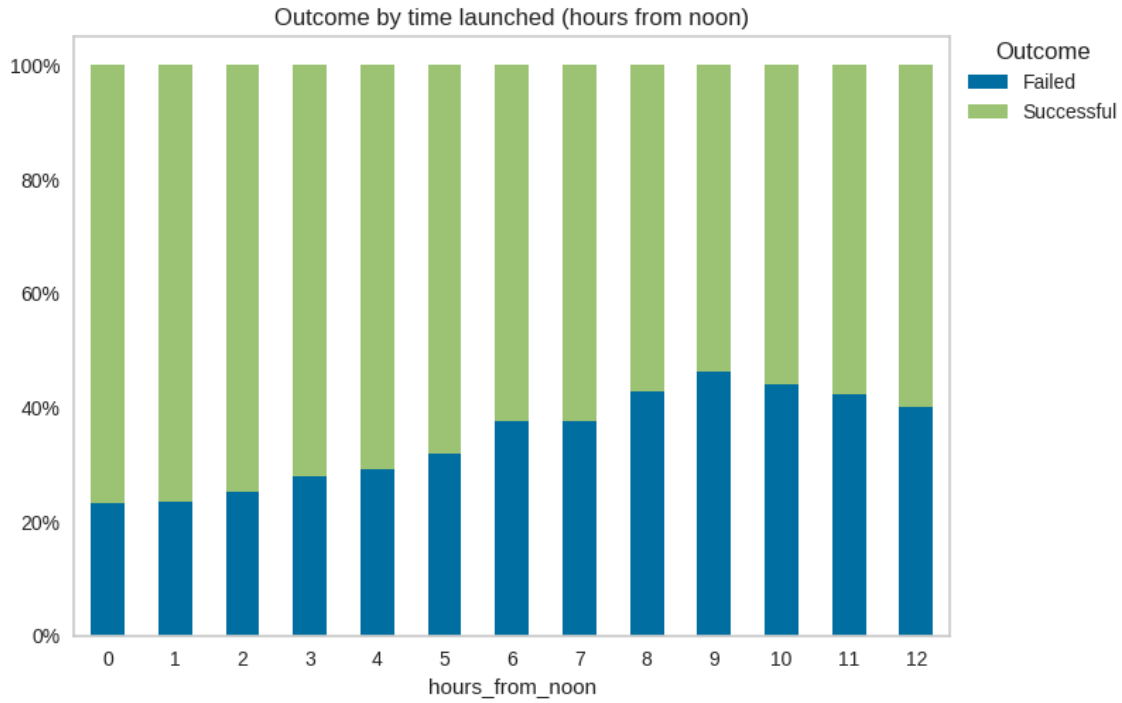
Project 2, Problem 1 Report
CIS 9660
Ben Attaway



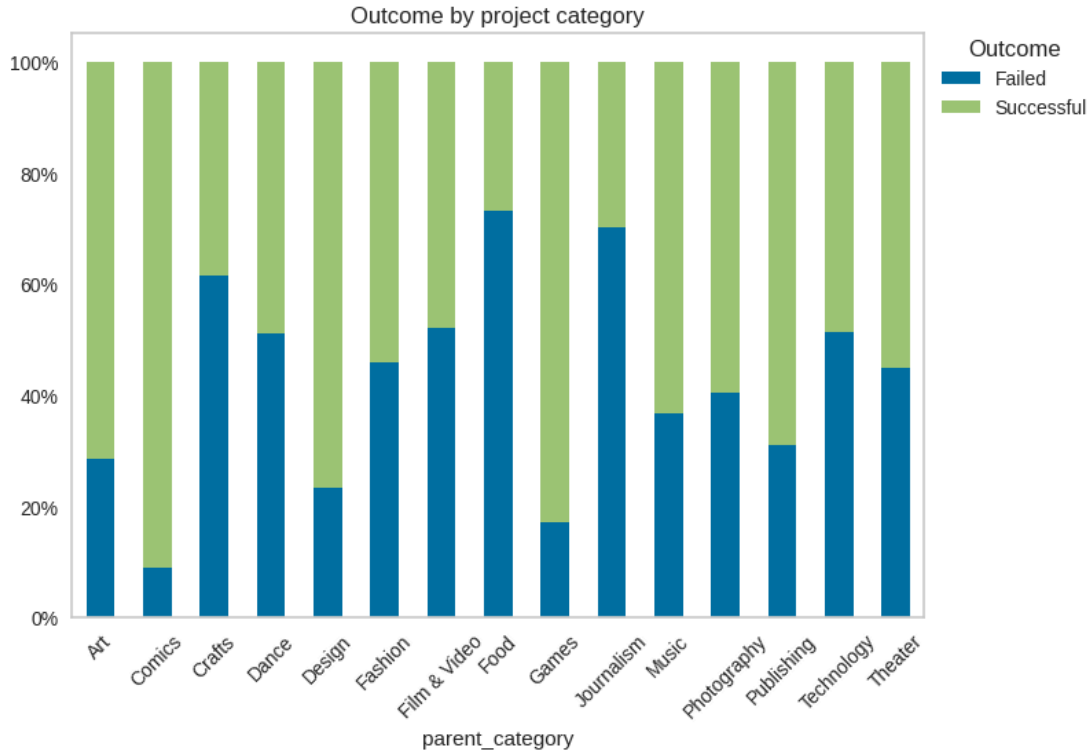


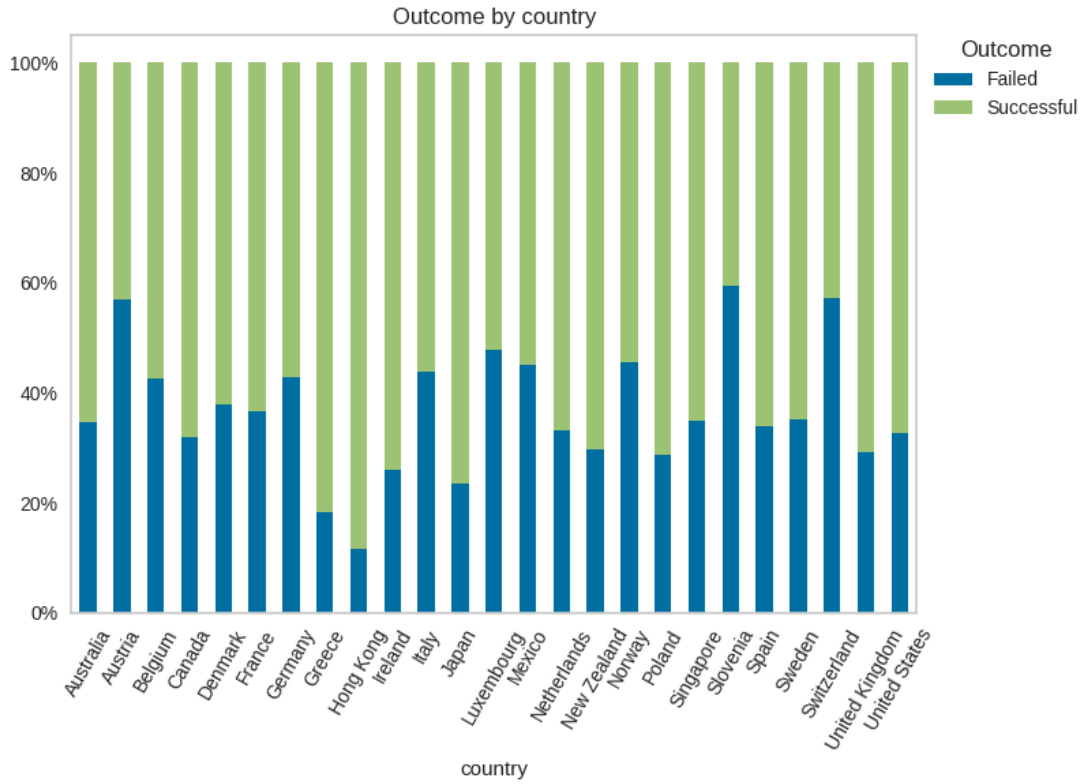
Project 2, Problem 1 Report
CIS 9660
Ben Attaway

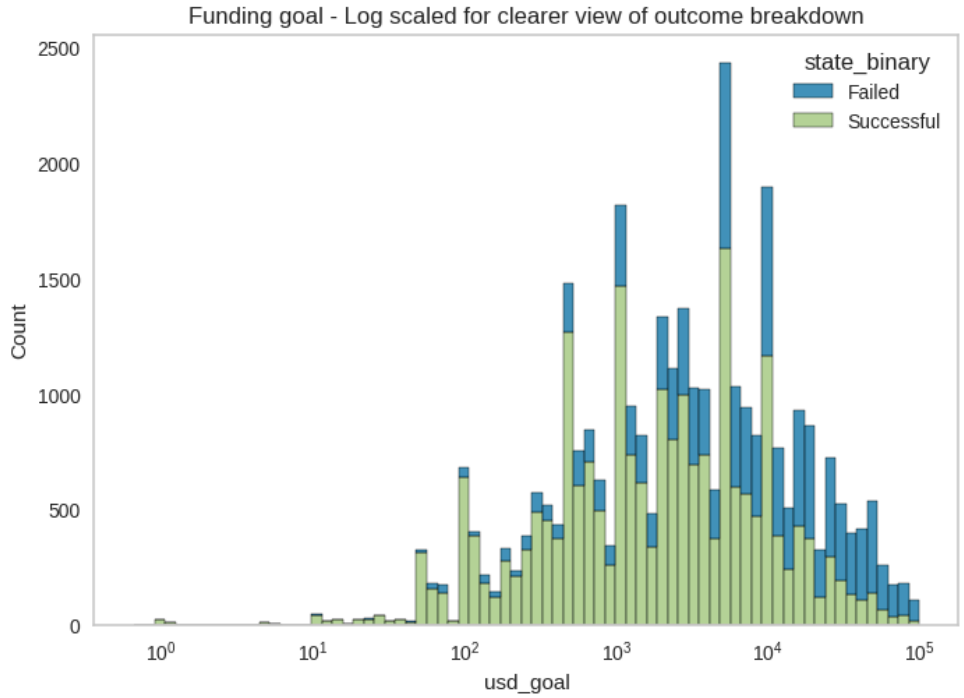




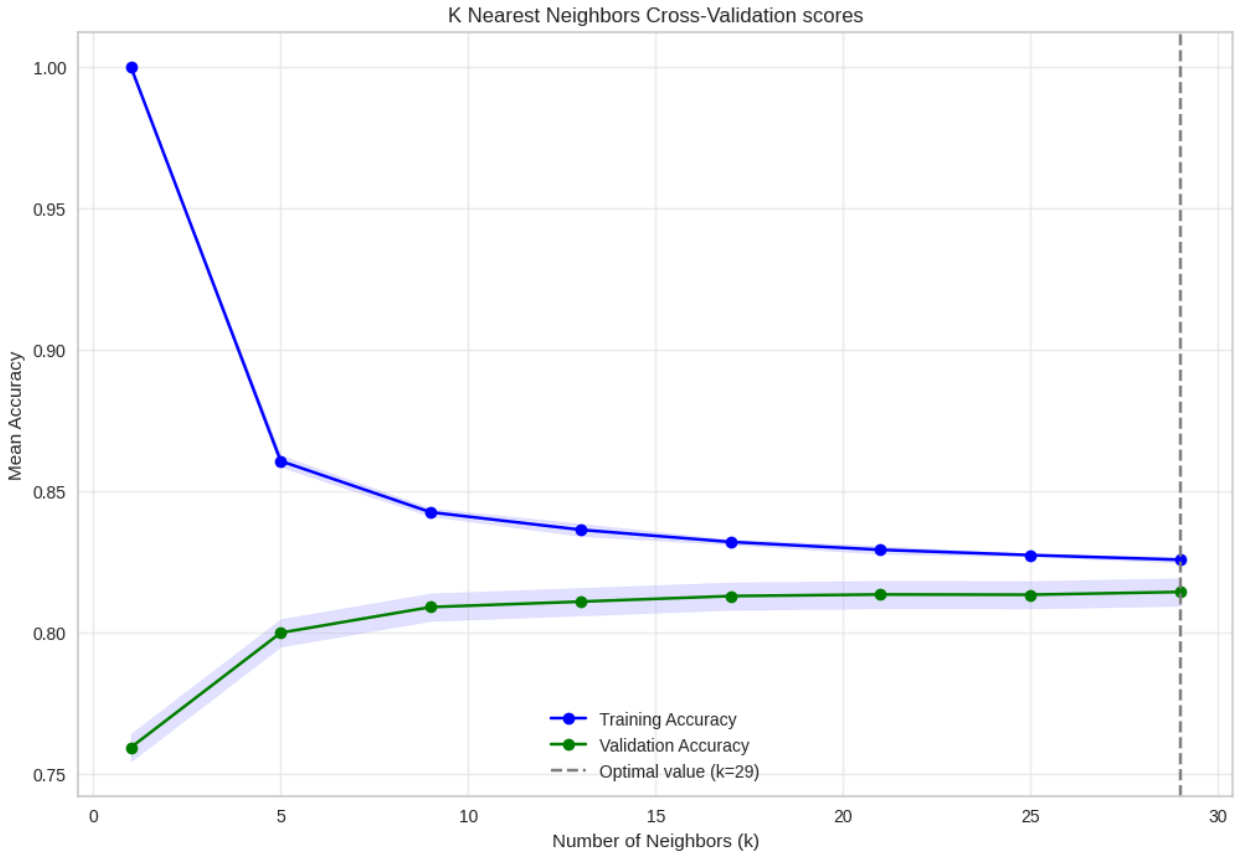
Project 2, Problem 1 Report
CIS 9660
Ben Attaway



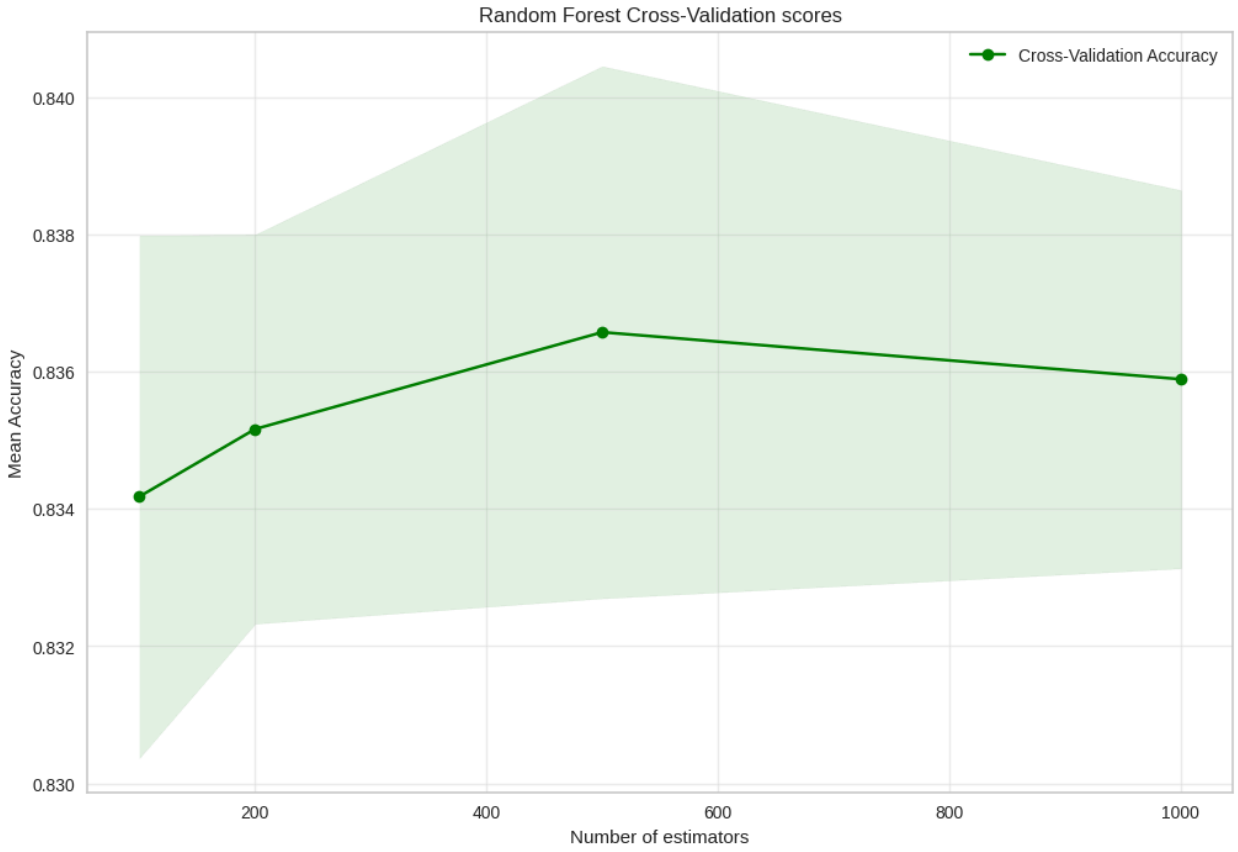




Model Performance Figures

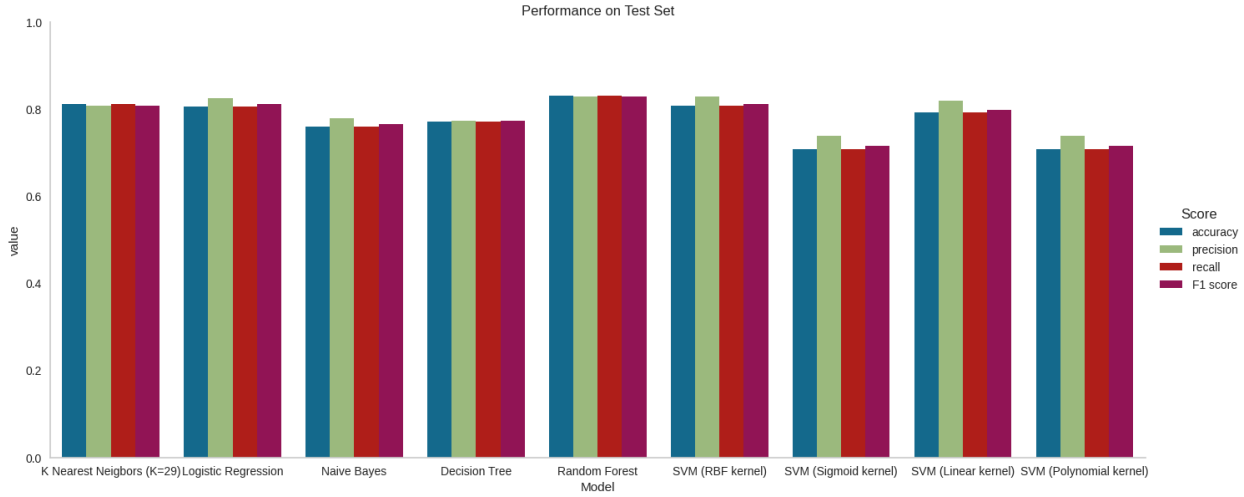


Project 2, Problem 1 Report
CIS 9660
Ben Attaway



(Larger versions of these two images are available in the /figures folder of the project repository and they are also in the Jupyter notebook.)

Project 2, Problem 1 Report
CIS 9660
Ben Attaway



Confusion matrices for Random Forest classifier (left) and SVM with RBF kernel (right)