Katie M. Chen
Mercedes Moore
Suramya Singh

Applied Machine Learning: Final Project Written Report

Introduction

Project Objective

This project aims to demonstrate each component of the machine learning process in both a fixed setting with a preemptively selected dataset and a dynamic setting in which web application users select their dataset and ML model of choice. The project tasks are outlined as follows: (1) data acquisition and pre-processing, (2) model building, and (3) web application development. For each task, this report details the steps taken to complete the respective task, explains the rationale for employed methodologies, provides code snippets and supporting graphics, analyzes the code's effectiveness and performance, and communicates the findings of this model experimentation. This report will consist of a statement of the project objective and a general introduction to the project, an analysis of the three tasks outlined prior, a summary of the project in totality, and a conclusion that reflects on the key learnings and takeaways from this endeavor.

General Overview

As an exploration into the abilities of machine learning models, this project employs a wide array of technologies and statistical methods to learn from data and make predictions. In particular, this project leverages the power of R, a versatile programming language commonly used for statistical computing and graphics. This project uses tools such as R code, RStudio, and RShiny to complete data preparation, model selection and training, model optimization, model evaluation, and interactive deployment. To facilitate collaboration amongst the project members, GitHub was used to account for version control and asynchronous work.

Completing tasks (1) and (2) outlined above required acquiring machine learning data. This was done by utilizing UC Irvine's Machine Learning Repository, a collection of databases, domain theories, and data generators commonly used by the machine learning community. Choosing an adequate dataset was a task that required ample consideration, as the project necessitated a dataset with high complexity and significant data quality challenges. Complexity was evaluated in terms of variety in feature type (numeric, character, categorical, binary, etc.), number of features, instances of data, and the perceived necessity of pre-processing. Examples of data quality challenges include missing data, erroneous entries, inconsistencies, skewed distributions amongst features, and similar difficulties.

The dataset that best fit these characteristics was the unnormalized version of the 2011 U.S. Communities and Crimes dataset. The dataset can be accessed through UCI's website here.

This dataset was an amalgamation of socio-economic data from the 1990 U.S. Census, law enforcement data from the 1990 Law Enforcement Management survey, and crime/law enforcement data from the FBI's 1995 Uniform Crime Reporting Program. The nature of the data created numerous complications in terms of the distribution of values and completion, as the LEMAS survey was limited to police departments with at least 100 officers (resulting in missing values and inconsistencies), the variable included multiple non-predictive variables, and omitted communities from certain states. The data was also extensive in size, with 125 predictive features, 4 non-predictive features, 18 potential goal features to choose from, and 2215 instances. Of the features, there was variety within feature type, as the dataset includes continuous and integer numeric features, categorical features, binary features, and In addition to this, the data was unnormalized, with the numerical data being input without any transformation. Due to these characteristics, the Communities and Crimes dataset was selected as the primary dataset to use for primary data analysis and regression modeling.

This dataset's target variables were continuous numeric values, necessitating the use of regression machine learning models and influencing the decision to make the RShiny web application strictly host regressive models. Performing pre-processing and data analysis, as outlined in the section below, gave further insight into the optimal model tactic as well as the best parameters for peak accuracy.

Constructing the models for the static dataset laid the foundation for the construction of the web application. These models were effectively generalized before being implemented into the RShiny platform to grant web users the ability to adjust model parameters and select their desired target and feature variables. Much like the static model experimentation, the RShiny web application allows users to change parameters to increase accuracy.

Overall, this project serves as a comprehensive exploration of the capabilities of machine learning models and demonstrates the power of data science. Though there are many aspects of applied machine learning were not included in this project for the sake of brevity, the project effectively offers a thorough examination of this course's topics and methodologies.

Data Analysis and Pre-Processing

Data Analysis

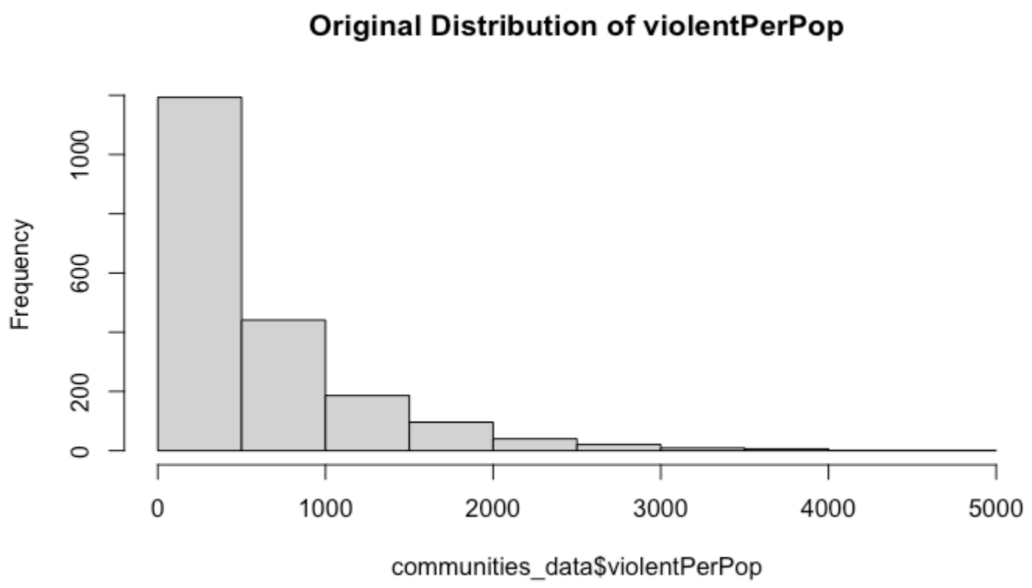
In addition to the preliminary data analysis conducted while selecting the dataset for this project, a thorough evaluation of the dataset's instances, feature variables, and statistical distributions was required. This evaluation was done to address many of the data quality issues identified earlier, as well as to create a solid foundation for model training and optimization.

Before modeling and observing data trends, the data was prepped to remove any inconsequential or missing data entries. This was done by (a) identifying missing values, (b) dropping columns with a disproportionate amount of missing values (the threshold for this was approximately 1800 missing values), (c) dropping any entries that had a missing value for our desired target variable (violentPerPop), (d) identifying feature variables with near-zero variances, and (e) dropping any non-predictive features. Code for these actions is contained in sections 3.A through 3.E in the attached RMarkdown file.

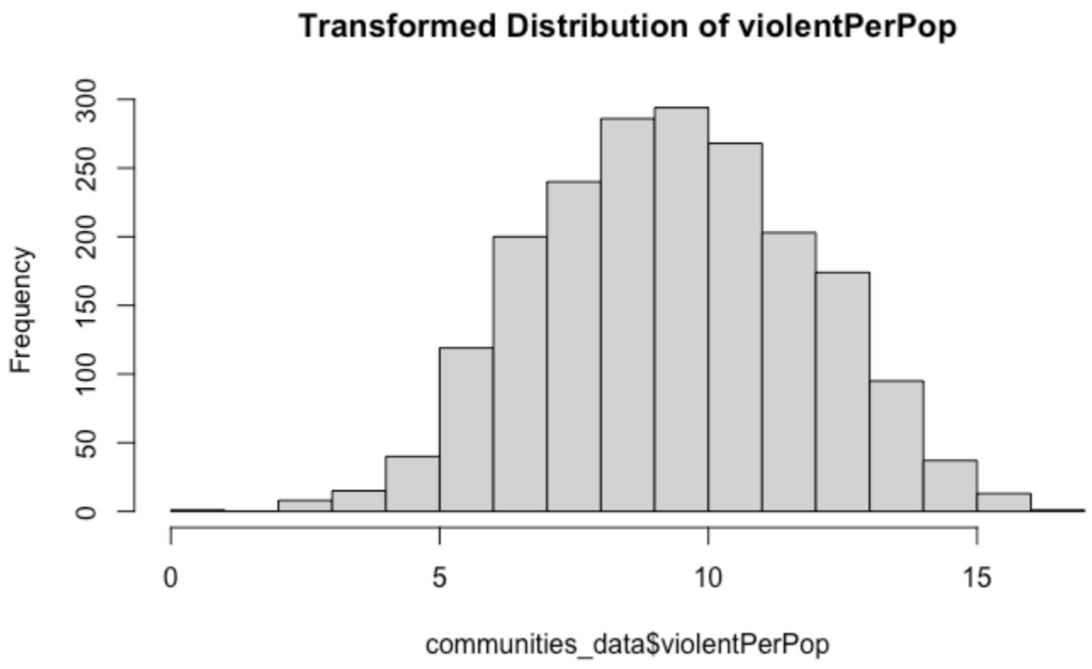
Once inconsistencies and missing values were accounted for, feature variable distributions were visualized using boxplots and histograms in sections 3.F and 3.G, respectively. Numerical variables were then grouped by the nature of their distributions:

- Notable Left Skew:
  - AsianPerCap, blackPerCap, fold, HispPerCap, HousVacant, indianPerCap, LandArea, LemasPctOfficDrugUn, numbUrban, NumIlleg, NumImmig, NumInShelters, NumStreet, NumUnderPov, OwnOccHiQuart, OwnOccLowQuart, OwnOccMedVal, PctForeignBorn, PctHousNoPhone, PctIlleg, PctImmigRec5, PctImmigRecent, PctLargHousFam, PctLargHouseOccup, PctLess9thGrade, PctNotSpeakEnglWell, PctPersDenseHous, PctPopUnderPov, PctRecentImmig, PctUnemployed, PctUsePubTrans, PctVacantBoarded, pctWFarmSelf, PctWOFullPlumb, pctWPubAsst, perCapInc, PopDens, population, racePctAsian, racepctblack, racePctHisp, ViolentCrimesPerPop
- Notable Right Skew:
  - PctBornSameState, PctFam2Par, PctHousOccup, PctKids2Par, PctSameCity85, PctSameState85, PctSpeakEnglOnly, PctYoungKids2Par, racePctWhite
- Considerable Outliers:
  - pctUrban

Perhaps the most notable skewed variable was the desired target variable, Violent CrimesPerPop (section 3.H)



Having an outcome variable with such a significant left-skew impacted the accuracy of any model used for the dataset. To address this, the outcome variable underwent a box-cox transformation in the pre-processing stage (3.I). The other skewed variables and variables with significant outliers also indicated that transformations and scaling would need to be implemented in the preprocessing for all features.



Another factor that was taken into account in the data analysis stage was features with near-zero variance amongst values. This was calculated in section 3.D, and revealed that the feature variable “pctPoliceAsian” had a near-zero variance.

The number of missing values throughout the dataset was also calculated (section 3.A), and revealed that a majority of the feature variables related to policing contained missing values. Though this was noted in the summary of the dataset provided on UCI, the volume of missing values for this subset of variables was much more significant than previously expected, with 1872 missing values for the following features:

numPolice, policePerPop, policeField, policeFieldPerPop, policeCalls, policCallPerPoop, policCallPerOffic, policePerPop2, racialMatch, pctPolicWhite, pctPolicBlack, pctPolicHisp, pctPolicAsian*, pctPolicMinority, officDrugUnits, numDIffDrugsSeiz, policAveOT, policCarsAvail, policOperBudget, pctPolicPatrol, gangUnit, pctOfficDrugUnit, policBudgetPerPop

Though dropping all rows with missing values for these columns was initially considered, doing so would result in losing almost 80% of instances. To remedy this, it was decided that these features would be excluded from the dataset, rather than the rows with missing values for these features (section. 3.B).

The final component of the data analysis process was measuring correlations between features and the desired target variable (section 3.I). This was done was to highlight which features were most important in determining the outcome variable, as to inform the creation of new feature variables and recognize which variables, if any, would be worth dropping. Correlation was gauged in two ways: individually and by grouped categories. Qualitative analysis of the feature variables revealed that the dataset's features could be grouped into the following categories: race/ethnicity, age, population, economy, education, employment, family, immigration, and housing. Though it was helpful to see each individual feature’s correlation, seeing the category’s correlation to the target was significantly more helpful, as it gave insight as to which subsets of features to pay attention to. Each category’s correlation was calculated by taking the magnitude of each feature’s correlation and then averaging these values (section 3.J). This process revealed that features in the race/ethnicity and family categories had the highest correlation with respect to the target variable (0.3181525, 0.4208915 respectively), while the age category had the lowest correlation (0.04606657); the remaining categories had a correlation magnitude between 0.1 - 0.3.

These analyses influenced the decision to implement the pre-processing steps outlined in the following section.

Pre-Processing

In response to the data trends highlighted in the data analysis, data pre-processing focused on erroneous/non-productive features and applying transformations. As noted earlier, there were a disproportionate amount of missing values in instances for features corresponding to policing, as well as one column with a near-zero variance. These columns were removed in code sections 3.B and 3.C.

To improve prediction accuracy, the target variable underwent a box-cox transformation to prevent further complications due to its left skew:

```
Java
shifted_variable <- communities_data$violentPerPop -
min(communities_data$violentPerPop) + 1  # Adding 1 to avoid zero
lm_model <- lm(shifted_variable ~ 1)
boxcox_result <- boxcox(lm_model)
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
communities_data$violentPerPop <- if (lambda == 0) log(shifted_variable) else
((shifted_variable^lambda - 1) / lambda)
```

As for the remaining transformations and traditional pre-processing steps (imputing remaining missing value, scaling numeric predictors, centering numeric predictors, etc), those actions were implemented using tidymodel's recipes package:

```
C/C++
blueprint <- recipe(violentPerPop ~ ., data = train) %>%
  step_string2factor(all_nominal_predictors()) %>%
  step_nzv(all_predictors()) %>%
  step_impute_knn(all_predictors()) %>%
  step_center(all_numeric_predictors()) %>%
  step_scale(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  prep()
```

Before applying pre-processing steps, the data was split into a 75/25 training/testing split (section 4.B), which was used throughout each of the five models tested once the recipe was to both the training and testing data:

```
C/C++
communities_train <- bake(blueprint, new_data = train)
communities_test <- bake(blueprint, new_data = test)
```

Model Selection, Training, & Optimization

Model Selection

As noted in earlier discussions regarding the dataset's nature, the dataset was limited to regression models. From the models covered in this course, it was determined that SVM, Random Forest, XGBoost, and ANN regression models would be the best models to predict the target feature.

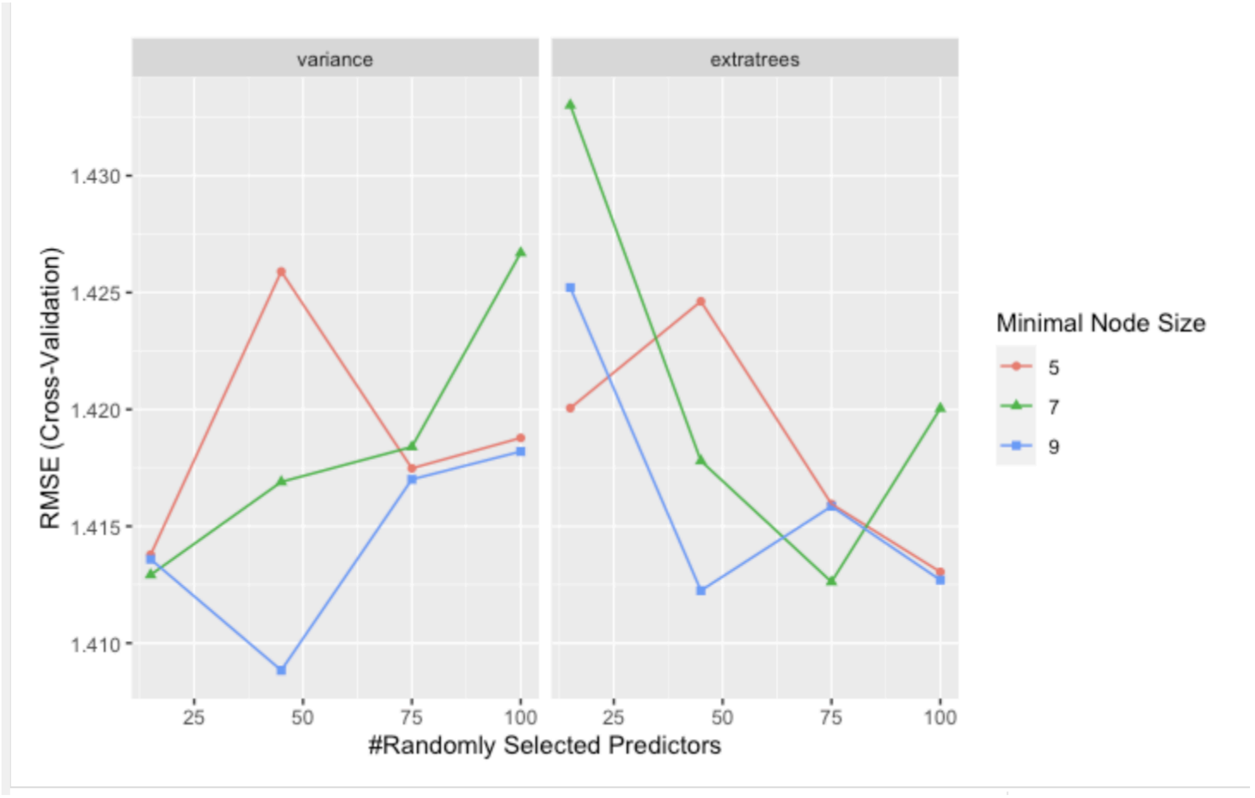
Each model had its own set of strengths and weaknesses. The support vector machine (SVM) for regression was selected due to its flexibility in its prediction boundary, as this model allows data points to be within the margin and has a good generalization performance. However, to accommodate for SVM's inability to compute predictions for datasets with missing values, many features, or larger datasets, the dataset had to impute missing values in pre-processing and the team had to account for long runtimes when modeling using SVM. As for the boosting model, XGBoost, this model was selected due to its ability to build a strong classifier, improved accuracy, and handling of imbalanced data (as our data varied widely amongst different metropolitan areas). Random Forest was selected for similar reasons as the other ensemble models but with the additional benefit of a lower model complexity and the bagging methodology. One shortcoming of the Random Forest model, however, was its selection of random variables, which made it possible for highly correlated variables to be left out. The ANN model was favored due to its ability to assign weight and bias to model input, an aspect that was very useful when dealing with variables with substantial variance in terms of feature-target correlation.

Each of these models used the same train/test subset of data and were evaluated using the root mean squared error.

Model Training & Optimization

Random Forest:

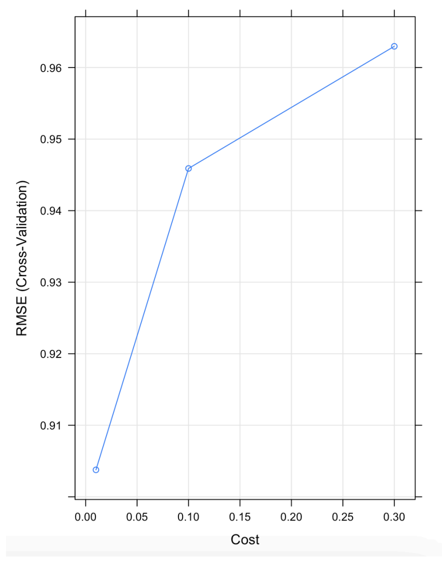
The Random Forest model was optimized by tuning the hyperparameters minimum node size, random variable try, and split rule. The basic random forest model revealed that error was minimized between 100 - 150 trees. The optimized model used a variance split rule with a minimal node size of 9 and 45 randomly selected variables and had a training RMSE of 0.6111782 and a testing RMSE of 1.37846.



Support Vector Machine (SVM):

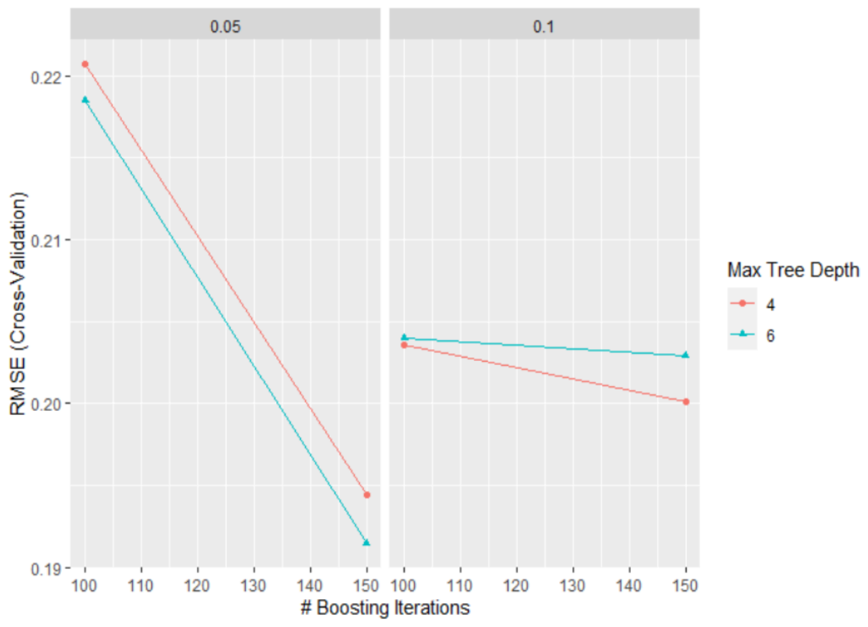
The SVM model was fine-tuned by optimizing parameters such as the kernel type, C value, and gamma coefficient for the 3 different SVM models (linear, polynomial, radial). After comparing the 3 models, the best performance was achieved with a linear basis function kernel, C=0.01. The final model with a linear kernel (C=0.01) resulted in an RMSE of 0.8509191 for the training data and 0.7952016 for the testing data.

Linear SVM



XGBoost:

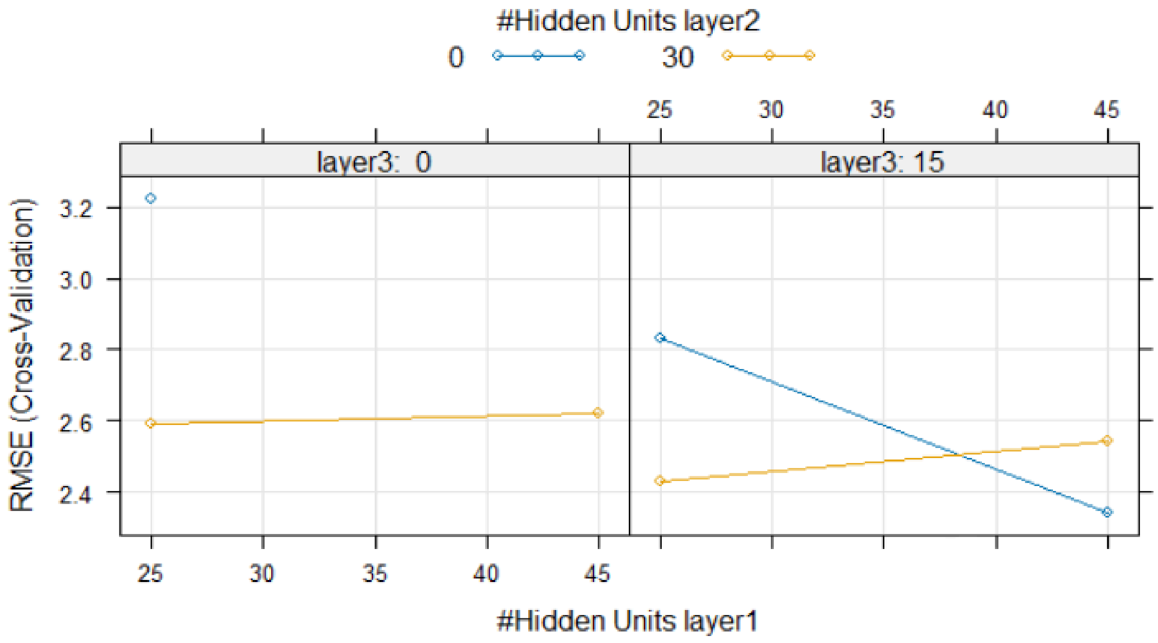
The XGBoost model underwent optimization by adjusting hyperparameters like learning rate, maximum depth, and number of trees. The regularization gamma was held constant at 0 and the number of features supplied is also held constant at 1. The preliminary XGBoost model suggested that the optimal performance was attained with a max tree depth of 6, an eta of 0.05, and 150 rounds. Following optimization, the model yielded an RMSE of 0.08 for the training data and 0.14 for the testing data.



Katie M. Chen
Mercedes Moore
Suramya Singh

Artificial Neural Network:

The Artificial Neural Network (ANN) model underwent optimization of three layers of a basic grid. The initial ANN model indicated that the optimal performance was achieved with 45, 0, and 15 units of each layer respectively. Following optimization, the model achieved an RMSE of 2.83 with the training data and 2.84 with the testing data.



Katie M. Chen
Mercedes Moore
Suramya Singh

Web Application

The web application is published here.

Upon completing these models for the static dataset, the team created generalized models to implement in a user-interactive Rshiny web application. This generalization was done so that users could upload any dataset and, given a numeric target variable, the web application could instantaneously train, optimize, and make predictions with any of the given models.

The Rshiny web application has 4 different sections: Uploading and Transforming Data, Data Visualization, Tuning Parameters and Model Optimization, and Developing Final Model.

Uploading and Transforming Data: The default dataset is the U.S. Crime Statistics dataset (communities_data_final.txt). Users can use this dataset or upload a dataset of their choosing.

After loading the data, users can select a response variable (numeric), training-test split ratio, and what preprocessing steps they want to perform on the dataset.

Data Visualization: Users can visualize the relationship between different categories in 3 different datasets (original, training, or testing) with either a scatter plot or histogram.

Tuning Parameters and Model Optimization: Users can choose between 3 different models (Random Forest, SVM, XGboost) and tune parameters to visualize which parameters have the lowest RMSE.

Developing Final Model: Users can then use the parameters they found in the previous section to create the most optimized model and see the results (RMSE and MAE).

Katie M. Chen
Mercedes Moore
Suramya Singh

Conclusion, Project Takeaways, & Future Iterations

This project has provided a comprehensive review and a modern-day example of the topics covered in Applied Machine learning. Both the analysis of the static dataset and the creation of the dynamic RShiny application provided insight into the modern machine learning progress, demonstrating the various strengths and weaknesses of different approaches, as well as their respective outcomes. Perhaps the most important takeaway from this project was the power of strategic selection & application of machine learning tools and models. Data analysis allowed project constituents to identify potential weaknesses before they impacted the accuracy of a model and indicated how different models could optimize predictions. In testing the assemble of selected models (RandomForest, SVM, XGBoost, and ANN), it was discovered that XGBoost yielded the most successful predictions, with the lowest RMSE value of any model tested. Upon further reflection, it appears that this was due to XGBoost's ability to fine-tune and generate multiple tree models to create an extremely fitted predictor model. Because there were so many feature variables in the selected dataset, it was imperative to maximize the accuracy of individual regression trees as to not allow for too large of a margin of error. This result, alongside the numerous lessons learned whilst creating a general web application, has helped stress the importance of mindful machine learning. Though it may be able to apply a model, good predictions only come when a data analyst has considered why they are applying a model and how that data may play to the strengths and weaknesses of the dataset they are analyzing. Machine learning is not merely a general algorithm, but an assortment of tools that can be strategically applied to perform succinct and accurate analyses.