Allstate Severity Claims Kaggle Competition

Team GrumpyKaggleKats

Luke Chu, Nelson Chen, Regan Yee, Cheryl Liu

kaggle

Allstate®
You're in good hands.

Roadmap

Summary and Goals

Pipeline

EDA and Preprocessing

Models

Results

Further Steps

Summary and Goals

- Dataset consists of 180k+ rows for training dataset
- Id column, 116 categorical features, 14* continuous features, 1 loss column
- No missing data - dataset is clean
- Features anonymized, continuous features scaled between 0 and 1
- Use these features to predict auto insurance claims

Attempt to build basic machine learning models on real world data and score high on Kaggle!

Roadmap

Summary and Goals

Pipeline

EDA and Preprocessing

Models

Results

Further Steps

Pipeline

1. Start off with applying traditional models to the training dataset
    a. **KFold validation**: train dataset split into Training and Validation data.
        i. **X_train**: Features of the training set
        ii. **Y_train**: Responses of the training set
        iii. **X_val**: Features of the validation set
        iv. **Y_val**: Responses of the validation set
    b. Train the model using the training set (X_train, Y_train)
    c. Get an average MAE of the cross-validated model
    d. Predict on the test set (test.csv)
2. Hypertune parameters and repeat!
3. Ensemble and/or stack and submit new model’s predictions to get sense of progress.

Roadmap

Summary and Goals

Pipeline

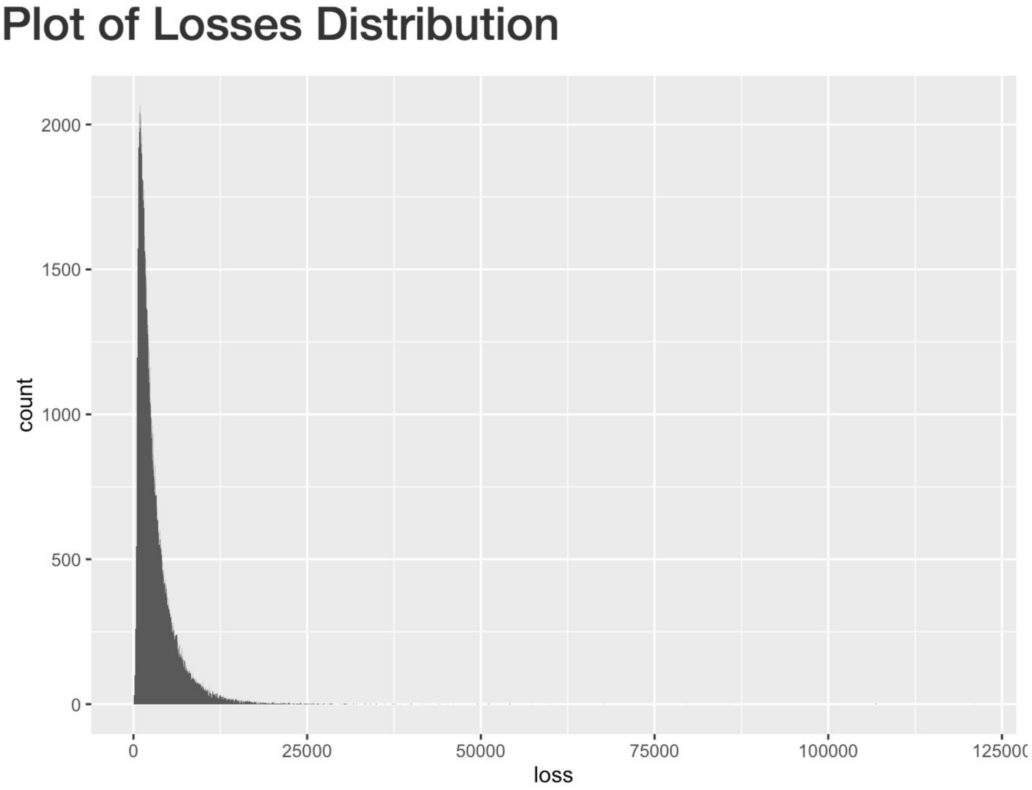
EDA and Preprocessing

Models

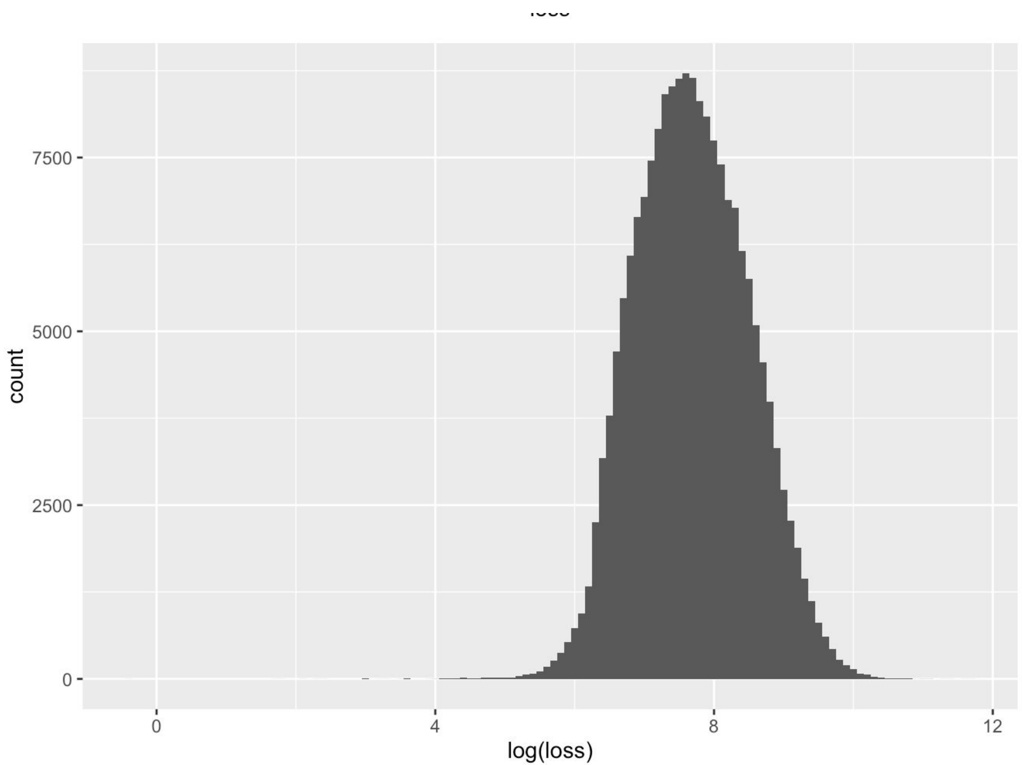
Results

Further Steps

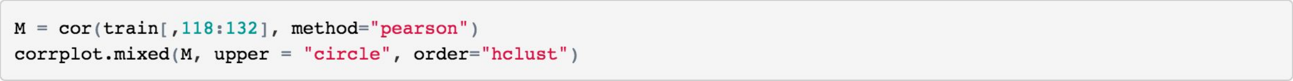
EDA - Distribution



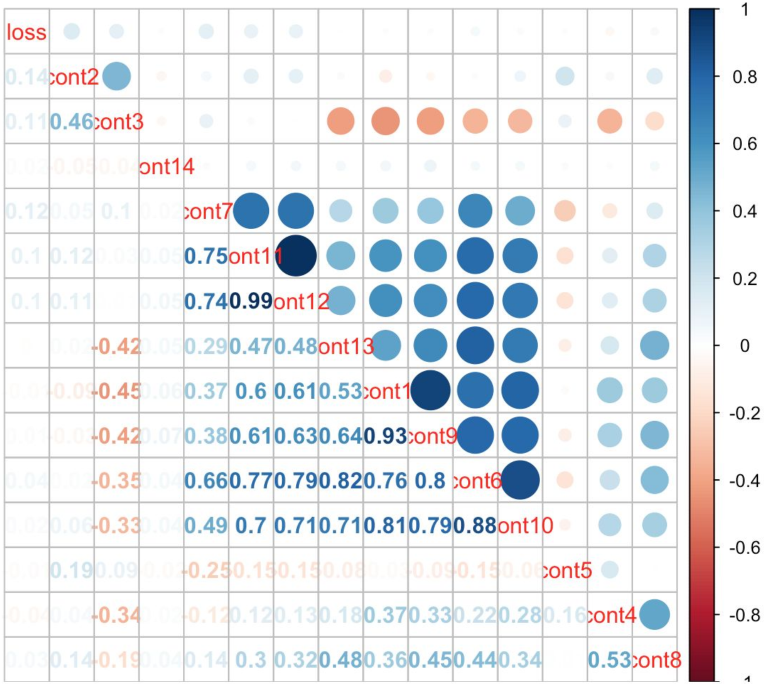
EDA - Distribution



EDA - Correlation Plot



```
M = cor(train[,118:132], method="pearson")
corrplot.mixed(M, upper = "circle", order="hclust")
```



EDA - Factor Analysis for Mixed Data

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	14.161089	1.3655824	1.365582
comp 2	7.430510	0.7165391	2.082121
comp 3	7.203723	0.6946695	2.776791
comp 4	7.026304	0.6775606	3.454352
comp 5	6.518144	0.6285578	4.082909

Roadmap

Summary and Goals

Pipeline

EDA and Preprocessing

Models

Results

Further Steps

Models - Lasso Regression

Data Preprocessing

Column Removed	Reason
id	An index, not a feature
cat112	States column
cont1	High correlation with cont9
cont6	High correlation with cont10
cont11	High correlation with cont12

Models - Lasso Regression

Data Preprocessing

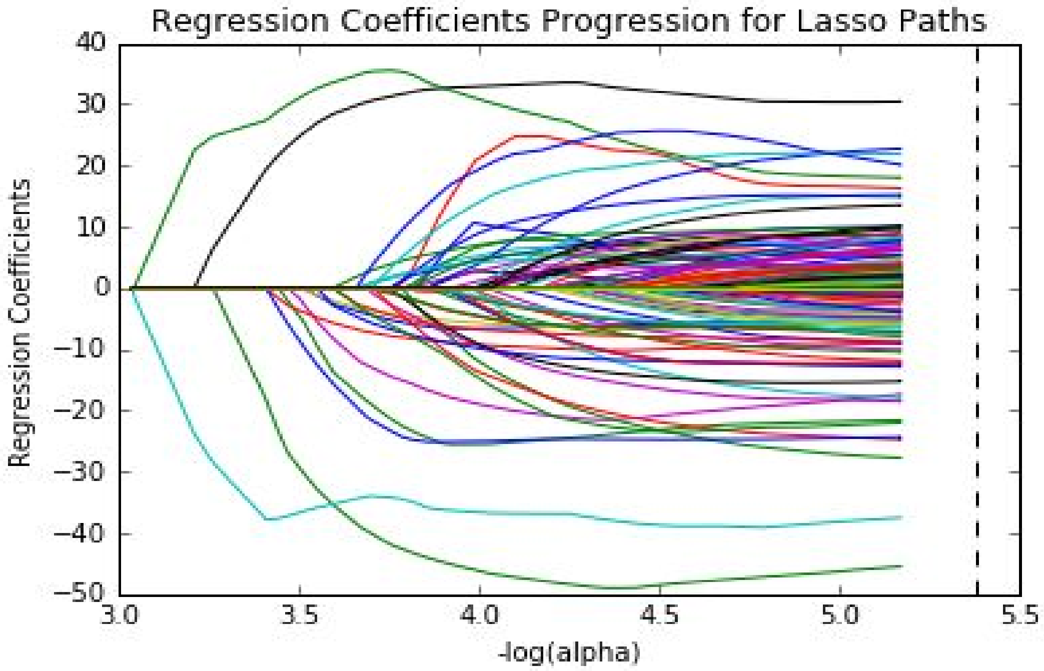
- Convert categorical data into numeric (get_dummies from pandas)

  The dummy dataset dimension: (188318, **127**) -> (150654,**1099**)

- Split the dataset (train.csv) into training and testing (8:2 ratio)
- Train the Lasso Regression model with cross validation value from 3--10
- After feature selection, only 326 columns kept

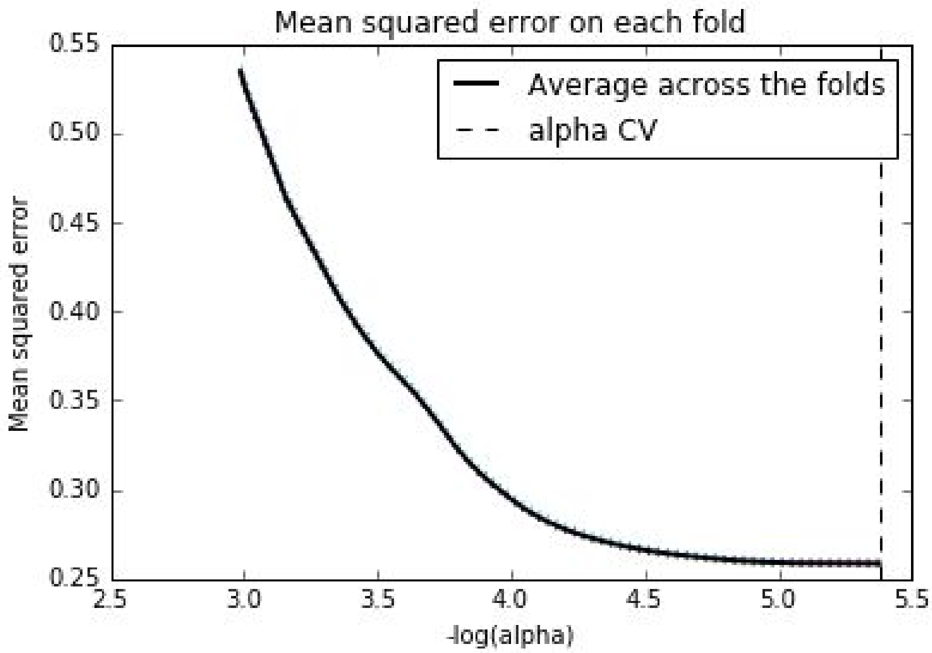
Models - Lasso Regression

- Coefficient Progression results



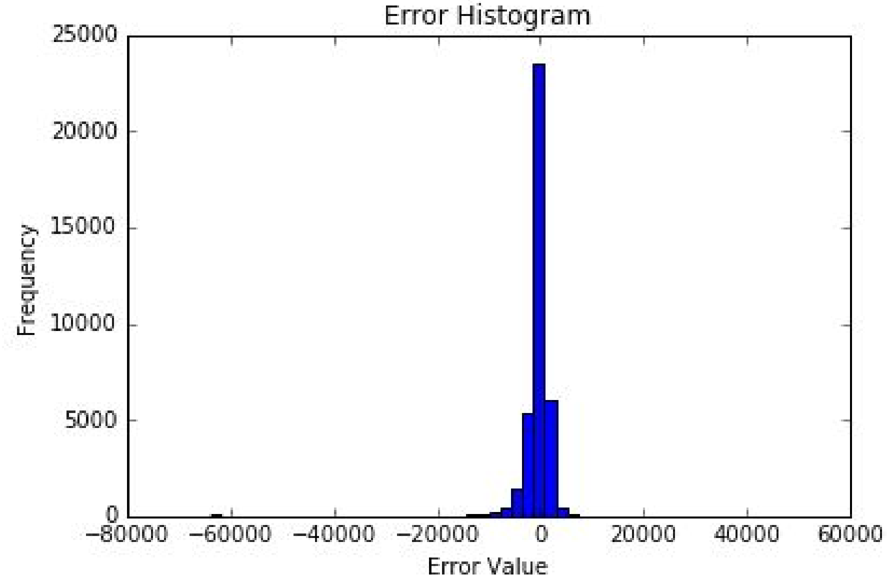
Models - Lasso Regression

- Mean squared error on each fold (5.3765326742844204)



Models - Lasso Regression

- Error distribution of Lasso Model on train.csv



- score on Kaggle: 1834.08715 on test.csv

Models - KNN

scikit-learn: KNeighborsRegressor library

- Tuning Parameter: K - number of nearest neighbors to consider
  - 425 - 440 neighbors for tuning
  - 425 gave best results in that range
- 10 folds created, 2 sampled for Cross-Validation

425-NN CV Score : 1721.150

425-NN LB Score: 1752.11803

Models - SVR

scikit-learn: SVR library

- Tuning Parameters:
- C - cost/penalty applied for making an error
    - Range over 10^ seq(-1,1,5)
- Epsilon - distance from margin such that penalty is zero
    - Range over 10^ [0.05, 0.01, 0.015, 0.02, 0.03, 0.1, 0.5]
- Gamma - variance of Gaussian - determines "influence" of support vectors
- 10 folds created, 2 sampled for Cross-Validation

Computationally Expensive! Results TBA

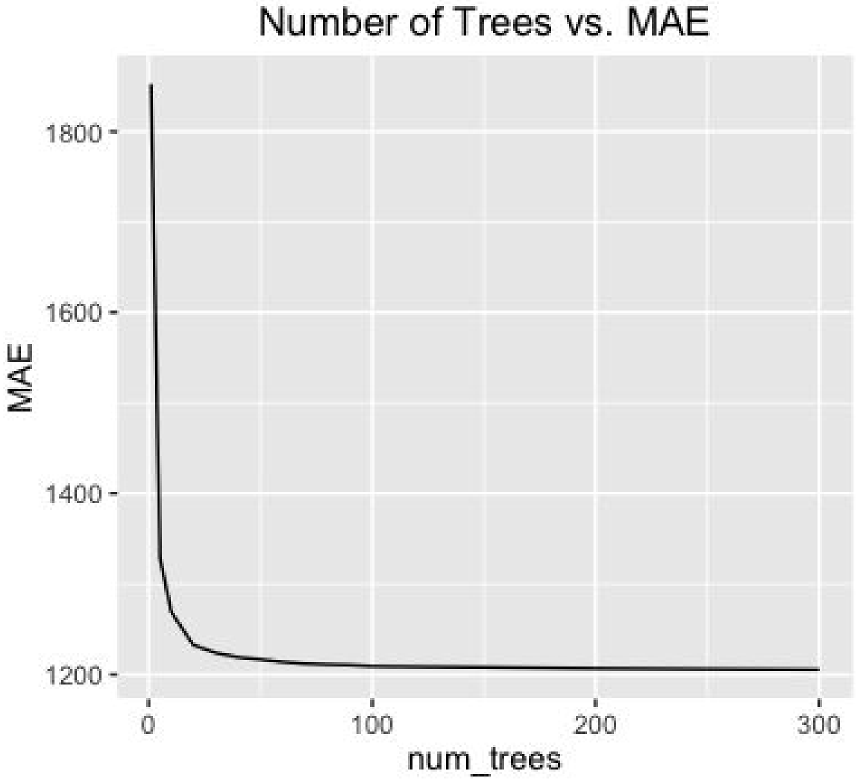
Models - Random Forest

scikit-learn: RandomForestRegressor library

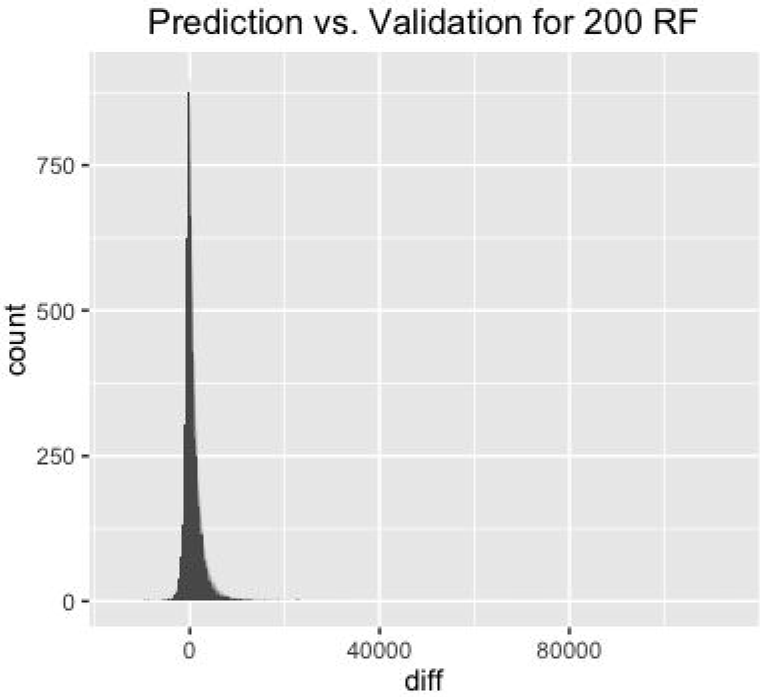
- Number of trees: [50,200]
- Max Features to Consider: sqrt(num_predictors) = sqrt(133) ≈ 11.53

50 RF CV Score: 1212.226	**50 RF LB Score:** 1188.92450
200 RF CV Score: 1202.28	**200 RF LB Score:** 1187.71536

Models - Random Forest - Number of Trees



Models - Random Forest -Prediction vs. Validation Set:



Models - Neural Network

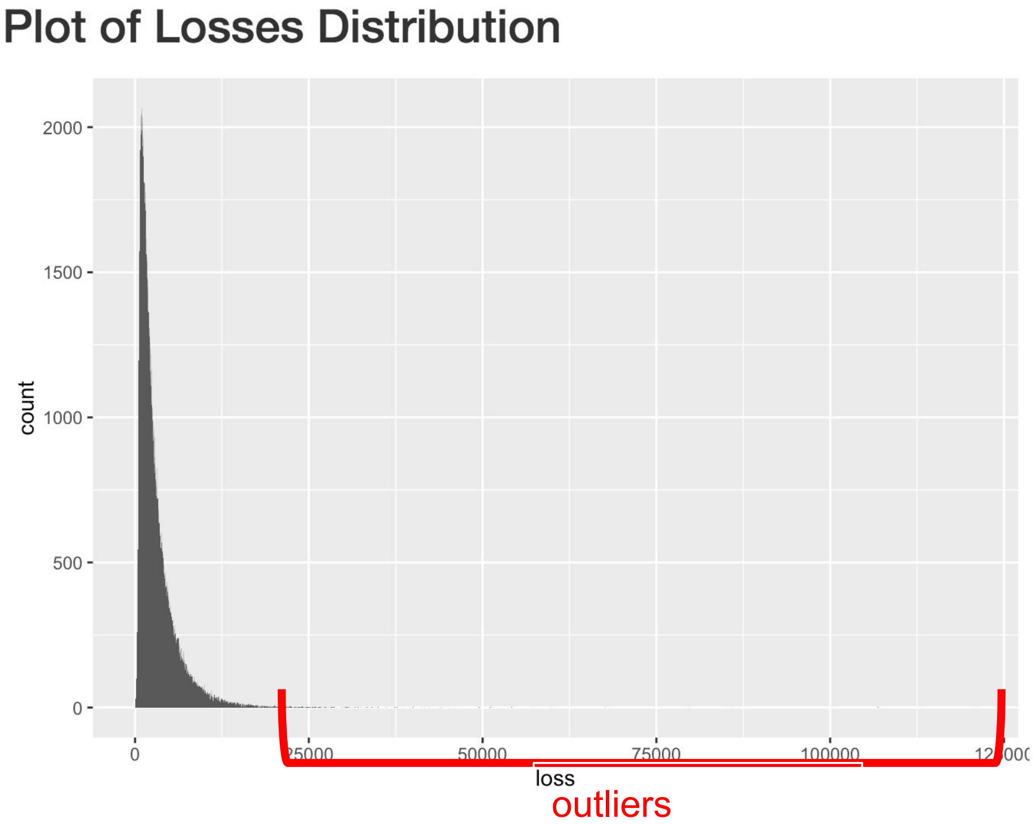
- Keras (Front-end) + Theano (Backend) python packages
- Single best model
  - Architecture : input → 400 Neurons → 200 Neurons → 50 Neurons → output
  - Regularized with dropout and batch normalization
  - 10 K-Fold with 10 bags per fold
  - Validation score: 1134
  - LB score: 1113
- Tried to add extra layers but could not reach minimal without smaller learning rate

XGBoost

- 10 K-Fold Cross Validation
- Parameters:
  - Learning rate: 0.01
  - Learning rate decay : 0.9996
  - Max trees = 10,000 with early stopping if validation score does not improve
- Validation score: 1133
- LB score: 1112
- Best single model

Outlier Analysis

- Ranges from 10340 to 121000
- Only 5315 pts out of 180k points
- 2.5% of the data but 90% of possible values
- Added feature in train set that signifies if it is an outlier
- Validation score: ~1050 with both xgboost and Neural Networks



Outlier Classification

- Goal: Train binary classifier to predict class on Test set
- Tried Logistic Regression -> 97.5% accuracy, 0.61 ROC AUC score
    - No better than just guessing all positive classes
- Tried XGBoost for classification while maximizing ROC AUC score
    - Most of the probabilities were near 0.5
    - Algorithm is very unsure of what was an outlier

Outlier Classification

- Used the Imbalanced dataset package in python to perform Synthetic Minority Oversampling Technique (SMOTE) + undersampling to generate new balanced dataset
- Ran another classifier with the results:
    - AUC of ROC: 0.8
    - Accuracy: 99.6%
- Great! Added new feature to test data and used XGBoost to run a regression: validation score: ~**1050** but LB: **1178**

Possible Reasons for Failure

- Leaderboard only displays scores on 30% of data, so 70% of data will need to deal with more outliers (unlikely but hopeful)
- Distribution for test set is different from the training set, so the classifier is not doing a good job
- Accidentally overfit to training set when we chose the cutoff based on loss variable (most likely)

Roadmap

Summary and Goals

Pipeline

EDA and Preprocessing

Models

Results

Further Steps

Results - Summary of Standalone Models

Kaggle Leaderboard Score:

- **Lasso Regression:** 1834.08715 (feature selection)
- **KNN Regression:** 1752.11803
- **SVR: ???**
- **Random Forest:** 50 Tree LB Score: 1188.92450

  200 Tree LB Score: 1187.71536

- **Neural Network:** 1113
- **XG Boost:** 1112

Results - Ensemble Models

- Average of two xgboost models and two Neural Network models
  - LB Score: 1108
- Used a optimizer to find best weights based on minimizing error of the validation sets to train loss
  - LB Score: 1105
- Simple weighted average where more weight was placed on the higher performing xgboost model and Neural Network model
  - LB Score: **1101**

Final Results and Insights

- Traditional models (linear regression, random forest, kNN) did not perform as strongly
- Single best model is XGBoost
- Runner up are Neural Networks
- But an ensemble of them is *even* better

Further Steps

- Perform more hyperparameter tuning to perfect single models
- Ensemble more models to correct mistakes
- Tune ensemble weights
- Explore stacking to further lower accuracy
- Examine outlier detection more closely, (i.e use anomaly detection algorithms instead of binary classification based on cutoff)

THANKS FOR LISTENING...