


Insurance Claim Predictions using Machine Learning

Presented by Henry, Harman, and Jeff

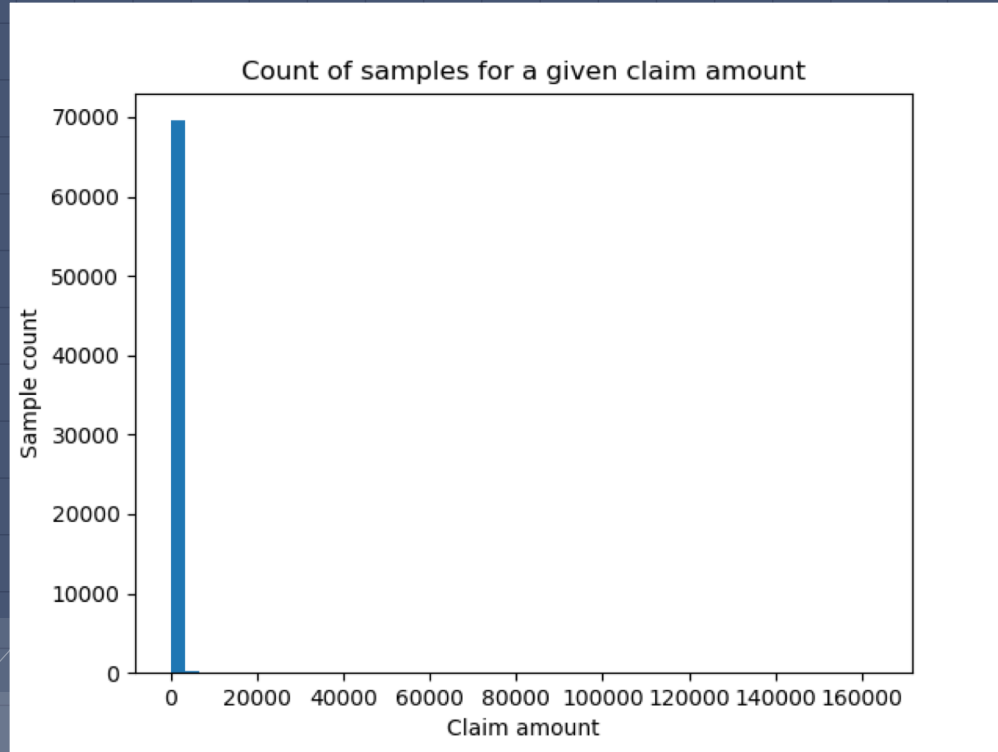
INTRODUCTION

- For a given person, on a given insurance term, they may, or may not, file a claim.
- Insurance companies are already able to predict, to a degree, the resulting claim amount.
- Improvements in prediction could allow companies to better adjust rates (ratemaking), saving money.

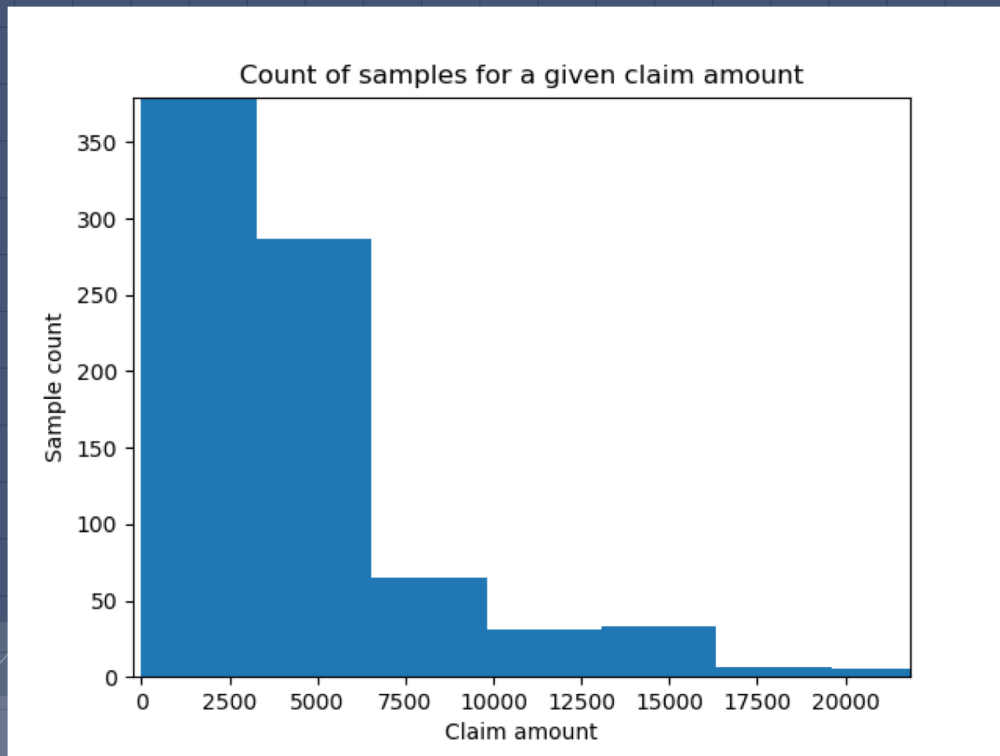


Is there a machine learning solution that will allow insurance companies to better predict claim amount?

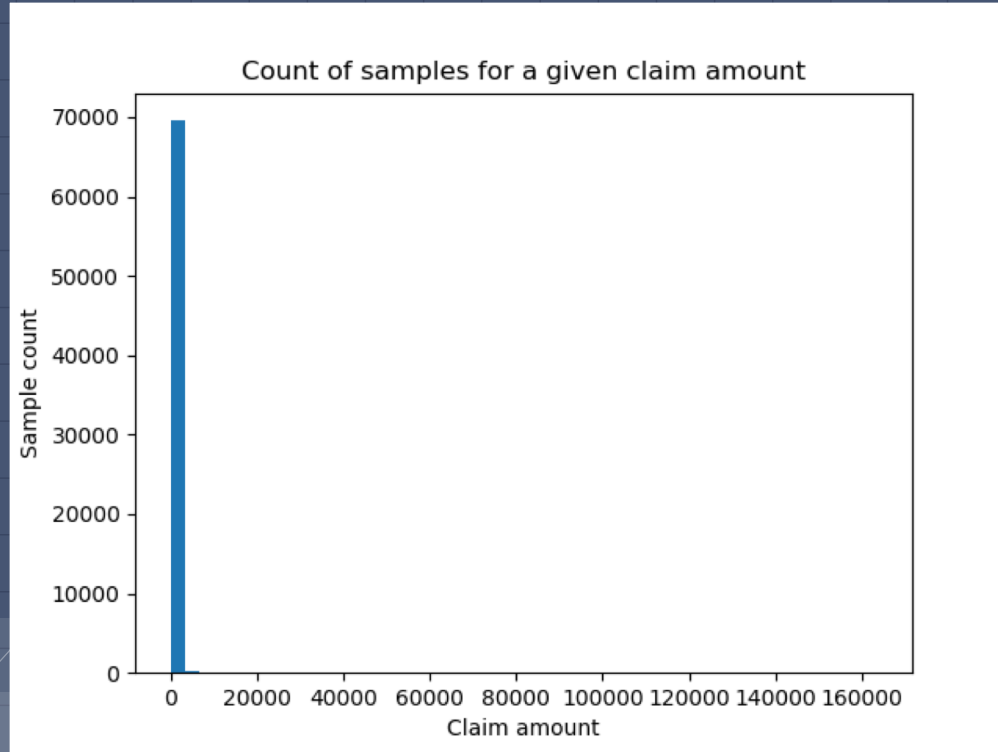
DATA EXPLORATION



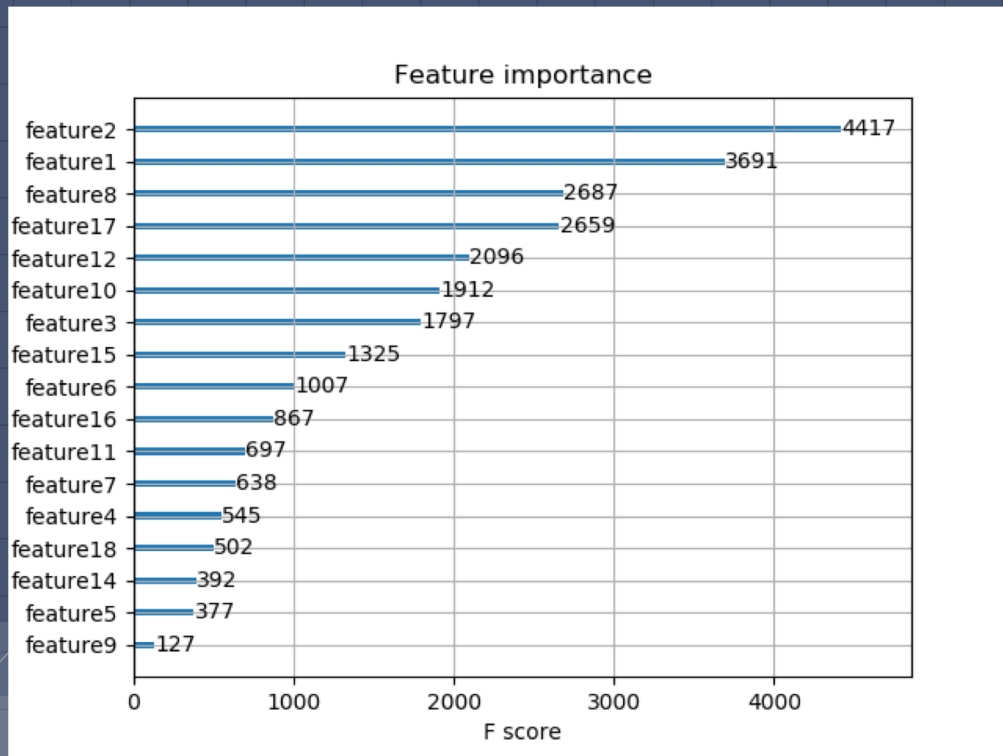
DATA EXPLORATION



DATA EXPLORATION



DATA EXPLORATION



SCORING

- We used two different scoring metrics: MAE and F1 Score.
- Mean Absolute Error (MAE) measures the average distance between a predicted and actual value.
- F1 Score measures accuracy using precision and recall.

$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

METHODS - 1st APPROACH

- Regression only (Linear, Polynomial, Random Forest)
- Results were poor (MAE was in the hundreds), due to there being many claim amounts with a value of 0, and some claim amounts resulting in a value that is in the thousands
- No way of predicting with regression alone whether a claim will result in a payout

CHECKPOINT 1 RESULTS

MODEL NAME	TEST SET MAE	TEST SET MAE POINTS	TEST SET F1 SCORE	TEST SET F1 SCORE POINTS
Random Forest	201.09	5	0.0916	7
Linear Regression	202.673	4	0.0916	7
Ridge Regression	2000.94	1	0.0916	7
Lasso	513.148	2	0.08828	4
All Zeros	107.036	29.5	0	2
Random Forest (AZ)	107.062	28	0	2

METHODS - 2nd APPROACH

- Add a classification model before regression
- Filter out non-claims
- Classifiers used:
 - Decision Trees
 - Logistic Regression
 - kNN
 - Random Forest
 - SVC
- Misleading accuracy

MISLEADING ACCURACY

- Using classification models, we got an initial accuracy of 95%
- Upon a closer look at the data, this result is very misleading
- The data is trained on mostly non-payout claims, so it predicted everything as being a non-payout claim



CHECKPOINT 2 RESULTS

MODEL NAME	TEST SET MAE	TEST SET MAE POINTS	TEST SET F1 SCORE	TEST SET F1 SCORE POINTS
Logistical Classification and Linear Regression	185.468	21	0.09149	5
	180.576	24	0.09235	14
KNN and Random Forest	113.431	48	0.10523	16
Random Forest and Random Forest	109.01	51	0.04898	4
SVC and Random Forest	164.069	25	0.13942	19
Decision Tree and Random Forest	109.172	50	0.04019	3

CHANGING THE SCORING METRIC / SAMPLING

- Instead of scoring on the overall prediction accuracy, have the model predict on only known claims
- In order to improve the accuracy of predicting claims, we changed the ratio of claims to non claims in the training
- A ratio of 50/50 resulted in a 72% accuracy when predicting on known claims, but still inaccurate. Model was now predicting everything as a claim

RANDOM FOREST - REGRESSOR

- Initial results returned Random Forest as the best regressor
- Stayed with Random Forest
 - Linear was 'untunable'
 - Polynomial was too computationally expensive
- Regressor was tuned using cross validation, scored on MAE



XGBOOST

- 'Regularized' Gradient Boosting
- Weak learners into strong learners
 - Initial weak model is improved by more models
 - More iterations, better 'accuracy'
- A more 'regularized model formation' to reduce overfitting

XGBOOST

- Hyperparameter tuning
 - Tuned 9 different parameters
 - e.g. max_depth, n_estimators, etc.
 - Model Selection with:
 - Random Search to find a range
 - Grid search to find the 'best' parameters
- Sampling ratio tuning
 - Increased back the number of non-claims
- Luck may have been a factor due to time

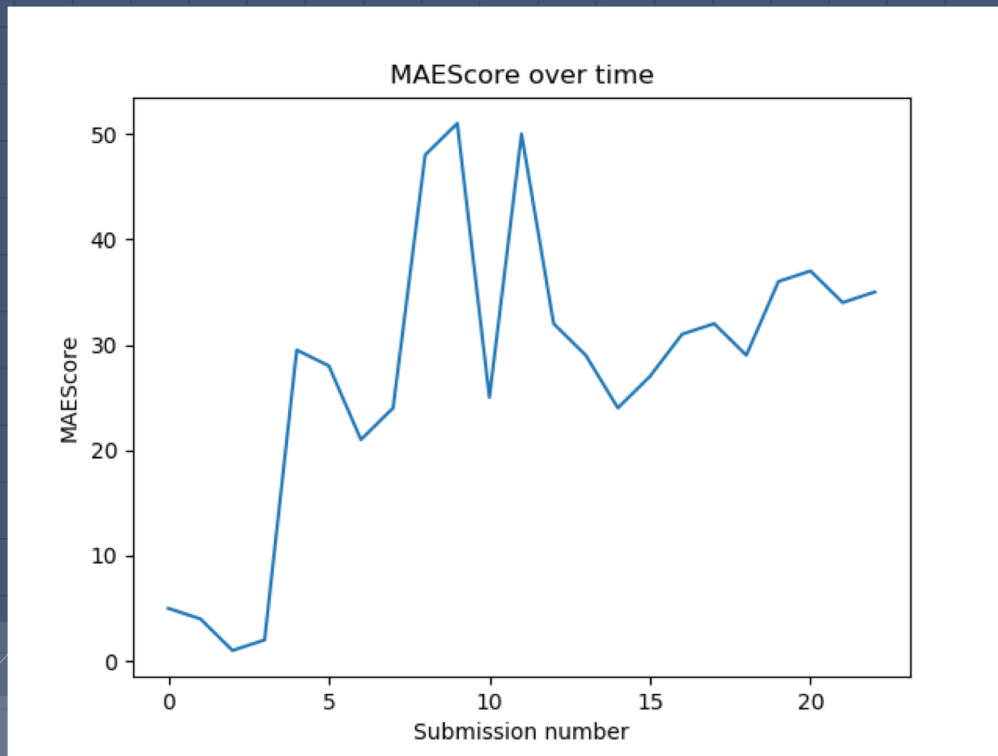
CHECKPOINT 3 RESULTS

MODEL NAME	TEST SET MAE	TEST SET MAE POINTS	TEST SET F1 SCORE	TEST SET F1 SCORE POINTS
XGBoost + Random Forest Regression	115.177	32	0.45731	19
	116.536	29	0.45223	17
	125.648	24	0.3165	9
	121.752	27	0.38317	11

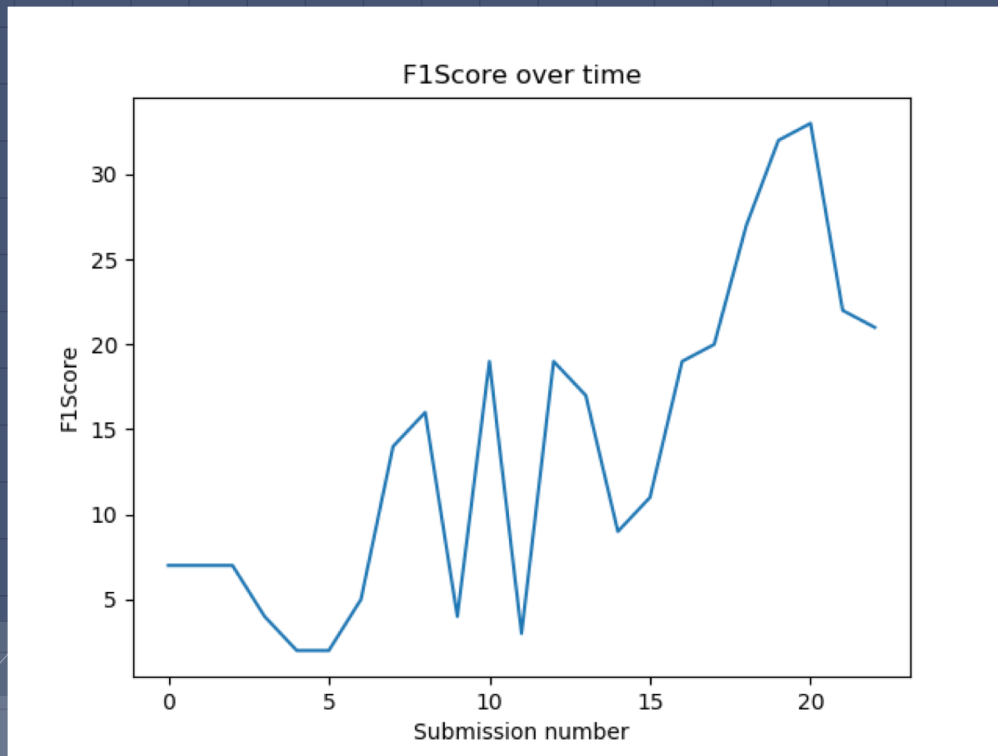
CHECKPOINT 4 RESULTS

MODEL NAME	TEST SET MAE	TEST SET MAE POINTS	TEST SET F1 SCORE	TEST SET F1 SCORE POINTS
XGBoost + Random Forest Regression	120.265	31	0.42918	19
	120.072	32	0.42993	20
	121.835	29	0.4599	27
	114.093	36	0.47669	32
	113.53	37	0.47935	33
Random Forest Classifier	116.533	34	0.43803	22
	115.628	35	0.43496	21

MAE SCORE OVER TIME



F1 SCORE OVER TIME



CONCLUSION AND INSIGHTS GAINED

- Zero-inflated data
- Classification and regression performed better than regression
- XGBoost was the best classifier
- Tuning of model parameters needed
- Adjustment of input ratios is necessary



PROJECT CONTRIBUTIONS

NAME	MODEL CONTRIBUTIONS	DOCUMENT CONTRIBUTIONS
Henry	XGBoost, Random Forest Classifier, kNN classifier, SVC, Decision Tree classifier, Random Forest Regressor, Model Tuning	Methods, Results
Harman	XGBoost, Linear Regression, Random Forest reg, Logistic Classifier, Random Forest Classifier, LGBM, Model Tuning	Methods, Classifier
Jeff	Linear & Polynomial Regression, Lasso, Ridge Regression, ZIP regression, Model Tuning	Introduction, Data Exploration, Results, Conclusion

REFERENCES

- Kuo, K., & Lupton, D. (n.d.). Towards Explainability of Machine Learning Models in Insurance Pricing. Retrieved from <https://kasaai.github.io/explain-ml-pricing/manuscript.pdf>.
- F1 score. (2019, November 30). Retrieved from https://en.wikipedia.org/wiki/F1_score.
- Brownlee, J. (2019, August 21). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved from <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>.
- Jain, A. (2019, September 13). Complete Guide to Parameter Tuning in XGBoost (with codes in Python). Retrieved from <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>.