PREDICTING CUSTOMER OVERSTRESS (KAGGLE COMPETITION)

Mohammad Tayyab Alam 24157

FINANCIAL DATA ANALYTICS

01 May 2025

Summary

The main aim of this project was to explore 11 different machine learning models to predict customer overstress.

Total Submissions ~ 72 by the name of Mohammad Tayyab Alam

Highest score of 0.95338 and a leaderboard position of 7 using XGBOOST model

I learned new concepts like Out of Fold predictions and stratified cross fold validations. This project also helped me appreciate and understand the importance of hyperparameter tuning and the need to reduce features to solve problems quickly. Feature engineering once again proved to be a crucial help.

Decision Tree

- **Best Accuracy: 0.51470 (Discarded)**
- **No. of Submissions: 3**
- **Average Run-time: 70 minutes**

Analysis

Used a basic decision tree model with hyperparameters tuned to reduce overfitting. The max depth was set to 5, and min_samples_split was adjusted to 2. The class imbalance was handled by balancing class weights.

Conclusion

- Performance: The model showed low performance and was not viable for the problem set.
- Advantages: Fast to run, easy to interpret.
- Disadvantages: Struggled with high-dimensional data and severe class imbalance; prone to overfitting without ensemble techniques.

KNN

- **Best Accuracy: 0.55477 (Discarded)**
- **No. of Submissions: 3**
- **Average Run-time: 5 minutes**

Analysis

RandomizedSearchCV was used to optimize the hyperparameters, particularly the number of neighbors (n_neighbors = 5). Despite using feature scaling, the model still struggled due to high-dimensional data and class imbalance. Hyperparameter tuning resulted in minimal improvements.

Conclusion

- Despite its simplicity and fast training time, KNN showed poor performance on this complex dataset. The lack of clear feature separability and sensitivity to noise led to suboptimal results.
- Given the weak performance and minimal improvements after tuning, KNN was discarded to divert focus towards better models like xgboost.

Logistic Regression

- **Best Score: 0.51836 (Discarded)**
- **No. of Submissions: 4**
- **Average Run-time: 10 minutes**

Analysis

Logistic regression was tested on the data with various configurations like missing value impute and scaling and then hyperparameter tuning via RandomizedSearchCV to find best parameters.

Conclusion

- Performance: Did not perform well in this dataset.
- Advantages: Easy to implement, interpretable.
- Disadvantages: Struggled with class imbalance and high-dimensional data, requiring significant feature engineering for improvement.
- Retried with feature engineering of sum and got ~ 0.72
- Parameters were C = 10

XGBOOST

- **Best Score: 0.95338 (Best Model)**
- **No. of Submissions: ~40**
- **Average Run-time: 45 minutes**

Analysis

RandomizedSearchCV was used to find the best parameters and then feature engineering was used to bump up score from 0.92 to 0.95338.

Conclusion

- Performance: High performance after significant hyperparameter tuning. Best overall model.
- Advantages: Robust, handles imbalanced data well, excellent with complex datasets.
- Disadvantages: Computationally intensive, sensitive to hyperparameter tuning.

CATBOOST

- **Best Score: ~0.94**
- **No. of Submissions: ~30**
- **Average Run-time: 1 hour**

Analysis

The model performed excellently but could not outperform XGBOOST. This model is better suited for categorical datasets but this data set was entirely numerical. Tuning resulted in minor improvements.

Conclusion

- Performance: Competitive but could not outperform XGBoost.
- Advantages: Handles categorical features naturally, robust to overfitting.
- Disadvantages: Slightly slower than XGBoost, marginal improvements with tuning.
- Parameters = iterations=6000, learning_rate=0.02, depth=1, l2_leaf_reg=3, border_count=32, loss_function='Logloss', eval_metric='AUC', scale_pos_weight=scale_pos_weight, random_state=42, thread_count=-1, verbose=False

LGBOOST

- **Best Score: ~0.93**
- **No. of Submissions: ~20**
- **Average Run-time: 30 minutes**

Analysis

LGBOOST was a strong performing model that trained quickly and feature engineering resulted in significant performance gains but it couldnt outperform catboost and xgboost. This model is suitable for time bound scenarios,

Conclusion

- Performance: Strong contender but underperformed relative to XGBoost and CatBoost.
- Advantages: Efficient and quick.
- Disadvantages: Less flexible than other boosting models.
- Parameters = n_estimators=1255, learning_rate=0.0096, num_leaves=121, max_depth=3, min_child_samples=97

Stacking (XGBoost, CatBoost, LGBoost)

- **Best Score: 0.95**
- **No. of Submissions: ~30**
- **Average Run-time: 90 minutes**

Analysis

This stacking model combined predictions of the top 3 performing models using the hyperparameters mentioned earlier. It tried to create the best version by having voting but it could not outperform xgboost.

Conclusion

- Performance: Worked well but could not surpass XGBoost. Complex ensemble method.
- Advantages: Combines strengths of multiple models.
- Disadvantages: High computational cost and complexity, requires careful tuning.

Naive Bayes

- **Best Score: ~0.5 (Discarded)**
- **No. of Submissions: 3**
- **Average Run-time: 20 minutes**

Analysis

- Naive Bayes performed poorly due to class imbalance and the high dimensionality of the dataset. Its assumption of feature independence is often violated in real-world datasets, which hindered its accuracy.
- Despite its simplicity and speed, it did not perform well on this complex problem, and its performance was far below other models in the experiment.

Conclusion

- Naive Bayes is suitable for small datasets as it is fast and easy to implement.
- However, a major flaw is that it assumes independence between features and this is problematic since it may not be true in high dimensional imbalanced datasets like the one in this project.
- It was discarded in favor of more complex and robust models like xgboost and catboost that handled the challenges of the dataset better.

Random Forest

- **Best Score: ~0.84**
- **No. of Submissions: ~14**
- **Average Run-time: 85 minutes**

Analysis

- Random Forest was the first successful model with decent performance, achieving an accuracy of ~0.84. It significantly outperformed simpler models like Decision Tree and Naive Bayes.
- The model reduced overfitting compared to a single decision tree by using an ensemble of trees, but was computationally expensive and slower than other models, especially for large datasets.

Conclusion

- Performance: First successful model with decent performance.
- Advantages: High accuracy, reduces overfitting compared to a single decision tree.
- Disadvantages: Computationally expensive, slower than other models for large datasets.
- Parameters = RandomForestClassifier(n_estimators=1000, max_depth=None, min_samples_split=5, min_samples_leaf=2, max_features=0.5, class_weight='balanced_subsample', random_state=42, n_jobs=-1, verbose=1)

Gradient Descent

- **Best Score: ~0.5 (Discarded)**
- **No. of Submissions: ~3**
- **Average Run-time: 30 minutes**

Analysis

- Gradient Descent had extremely poor performance and was therefore not suitable for this problem. It is more suited for small datasets.
- The model was tested with 200 estimators and worked well on small data sets but not large.

Conclusion

- The model was discarded due to low performance on the dataset.. The problem at hand requires a faster learning enviroment
- Models inability to perform well on a complex data set made it unsuitable for this project..

ADABOOST

- **Best Score: 0.52 (Discarded)**
- **No. of Submissions: ~4**
- **Average Run-time: 30 minutes**

Analysis

- AdaBoost showed low performance and was discarded due to inefficiency in this context. The model struggled with class imbalance and did not scale well with the large feature sets in the dataset.
- Despite being simple and effective for smaller datasets, it failed to deliver competitive results In this particular problem.

Conclusion

- AdaBoost was discarded as it did not improve significantly with hyperparameter tuning.
- While it is effective for smaller datasets, it struggled to handle the challenges of the large and more complex data set in this project.
- Its inability to address class imbalance and handle large feature sets limited its potential in this project.

Thank you!

Mohammad Tayyab Alam 24157

01 - 05 - 2025