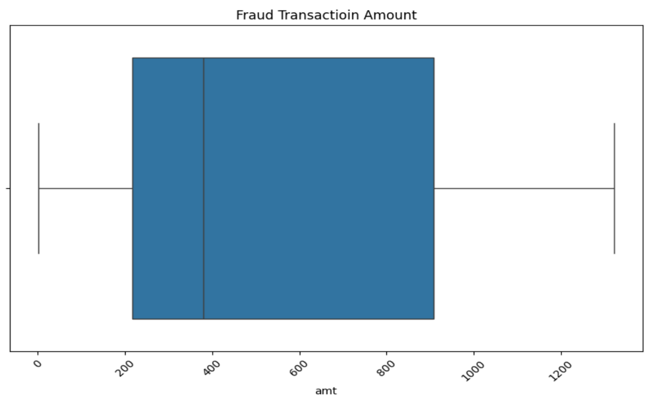
Name: **Youxuan Ma, U23330522**

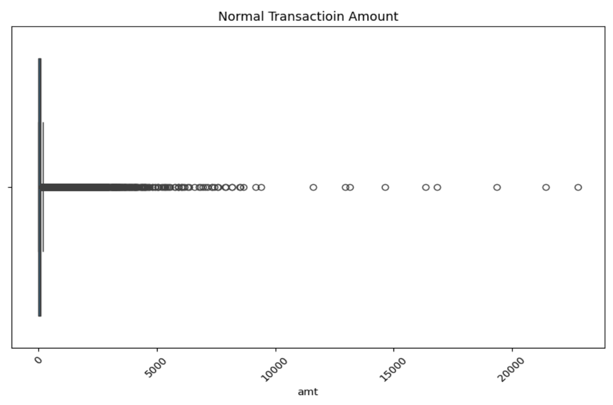
Kaggle Username: **MarkMa316**

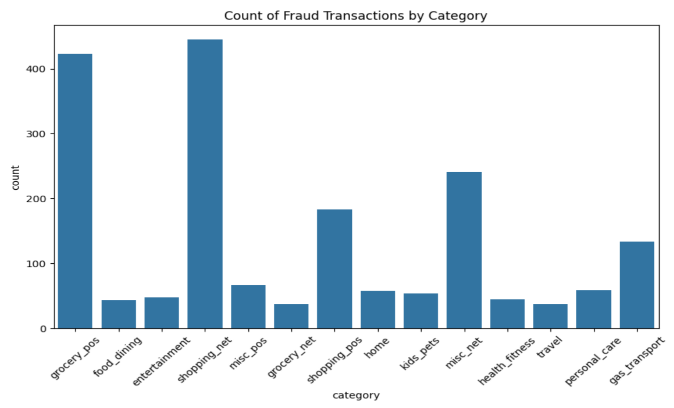
Midterm Kaggle Competition Report

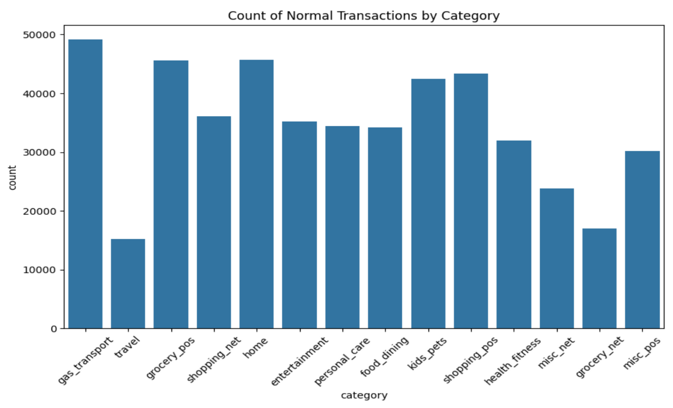
1. Data Analysis and Key Insights

My initial step was to conduct thorough data exploration and analysis, aiming to understand the nuances of credit card transactions and identify potential indicators of fraud.
Below are some of the graphs I plotted during the process:









Analyzing these plots and applying more statistics, I discovered that there are some clear patterns between the fraudulent and normal transactions. For instances:

- Fraudulent transaction amount has a mean of \$530.27, with very few outliers and \$1320.92 maximum, whereas normal transaction amount has a mean of \$67.50, yet it's highly skewed and has a maximum of \$22768.11.
- In addition, fraudulent transactions mostly centered around 4 categories: `grocery_pos`, `shopping_net`, `shopping_pos`, and `misc_net`. In contrast, normal transactions spread across all categories relatively uniformly, only with small exceptions of `travel` and `grocery_net` categories.
- Moreover, fraudulent transactions often occurred at odd hours, usually after 10 pm and before 6 am.
- Also, geographically, fraudulent transactions usually happened at locations significantly distant from the user's typical transaction spots.

2. Feature Engineering and Selection

With these insights and analyses, I then carefully engineered existing features and created several new ones to better capture the transaction patterns, user behavior, and contextual anomalies:

- `*Transaction Amount Relative to User Average*`: Highlighted transactions that deviated significantly from a user's typical spending behavior.
- `*Time Since Last Transaction*`: Detected unusually rapid succession of transactions, possibly indicative of fraud.
- `*Distance from Home to Merchant*`: Utilized geographical distance as a proxy for anomalous transactions outside a user's typical transaction locale.
- `*Distance to Previous Transaction by the Same User*`: Measured the geographical distance between a user's current transaction and their previous one, as significant geographical jumps between transactions, especially within short time frames, could indicate fraudulent activity.
- `*Transaction Location Consistency*`: assessed the consistency of transaction locations for each user.
- `*Fraud Rate per Merchant and per Category*`: Calculated the fraud rate (the proportion of fraudulent transactions) for each merchant and category, providing a risk profile based on historical data.
- `*Transaction Amount to `Fraud` and `Not Fraud` Similarity*`: Compared the transaction amount to typical amounts for fraudulent and legitimate transactions.
- *`Transaction Amount Anomaly Score per Category and per Merchant*`: Calculated an anomaly score to each transaction based on how far its amount deviates from the expected range for its category and merchant.

3. Model Selection and Training

Given the binary classification nature of our competition, I experimented with several models: Logistic Regression, Random Forest, XGBoost, and ensemble techniques of Bagging and Voting Classifiers.

- At last, I chose the Random Forest and XGBoost models due to their high F1-scores among others.
- Then, I used the best of them trained, a XGBoost model, to make the Bagging Classifier.
- Finally, I ensembled a Voting Classifier that contains all the three models trained.

- *Random Forest* is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. It corrects for decision trees' habit of overfitting to their training set.
- *XGBoost* stands for eXtreme Gradient Boosting. It is an implementation of gradient boosted decision trees designed for speed and performance. It builds the base models sequentially, with each new model correcting errors made by the previous models.
- *Bagging Classifier* involves training the same algorithm on different subsets of the training data, then aggregating their predictions. This approach reduces variance and helps avoid overfitting.
- *Voting Classifier* combines the predictions from multiple different models.

- I employed Bagging with XGBoost as base estimator to fully leverage its individual strengths while mitigating its tendency to overfit.
- I configured the Voting classifier for soft voting to integrate the probabilistic insights from the three best models I trained: the Random Forest, XGBoost, and Bagging Classifier, aiming for a balanced decision based on the confidence level of each model.

4. Tuning Model Parameters

I employed grid search and cross-validation to fine-tune the hyper-parameters — learning rate, max depth, n_estimators, and more — for the Random Forest and XGBoost models, optimizing for the F1 score to balance precision and recall effectively.

5. Validation Strategy

- Due to the expensive computing nature of my models, I used Hold-out validation for each of the individual models, allocating 80% of the data to training and 20% to testing. This approach provided me with a quick and straightforward way to estimate my model performance.
- Then in the end, for the final Voting classifier I ensembled, I employed Stratified K-Fold validation, which is much more comprehensive yet expensive. It allowed me to ensure that each fold had a proportionate representation of fraudulent and non-fraudulent transactions, which is especially important in an imbalanced datasets like ours.

6. Challenges Encountered

- *Data Imbalance*: A significant challenge was the imbalanced nature of the dataset, with fraudulent transactions being much rarer than legitimate ones. I addressed this by carefully tuning my models to focus on the F1 score, and applying stratified validation strategies.
- *Feature Selection*: Identifying the most predictive features while avoiding overfitting required immense efforts. I have experimented with large number of different combinations of features and models and their hyper-parameters to arrive at the optimal result I can get with my best efforts.
- *Hyper-parameter Tuning*: The vast hyper-parameter space of Random Forest and XGBoost presented a challenge, requiring extensive computational resources for grid search. I mitigated this by employing a more targeted search space based on my experiences and preliminary analyses.