Ethereum Fraud Detection

Romil Patel

1. Feature lists

a) **Manual Selection:** *feat_manual_subset.csv*

This subset was created by manually going through the dataset and dropping columns with only 0 values, as such columns contribute no additional information for the model and can add unnecessary complexity. This helps reduce the feature space without losing valuable data.

Selected features: ['a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'a11', 'a12', 'a13', 'a14', 'a18', 'a19', 'a20', 'a22', 'a23', 'a24', 'a25', 'a26', 'a27', 'a28', 'a29', 'a30', 'a35', 'a36', 'a37', 'a38', 'a39', 'a40', 'a44', 'a45', 'a46', 'a47']

b) **Information Gain-based Selection**: *feat_info_gain_subset.csv*

Using mutual information criterion, the **top 25** attributes with the highest information gain relative to the target were selected. This method identifies features with significant predictive power.

Selected features: ['a3', 'a7', 'a10', 'a11', 'a18', 'a20', 'a22', 'a23', 'a24', 'a25', 'a26', 'a27', 'a28', 'a29', 'a30', 'a35', 'a36', 'a37', 'a38', 'a39', 'a40', 'a44', 'a45', 'a46', 'a47']

c) **Correlation-based Selection**: *feat_correlation_subset.csv*

Features were selected based on their correlation with the target variable. The **top 20** correlated features were included in this subset.

Selected features: ['a47', 'a3', 'a2', 'a18', 'a5', 'a4', 'a46', 'a14', 'a8', 'a7', 'a1', 'a13', 'a9', 'a23', 'a38', 'a40', 'a39', 'a25', 'a10', 'a27']

2. Classifier details and results

a) **XGB:** The XGB Classifier from the xgboost library is an implementation of the XGBoost algorithm tailored for classification tasks. XGBoost, short for "Extreme Gradient Boosting," is an advanced ensemble learning technique that combines the predictions of multiple decision trees to improve accuracy and robustness.

- **n_estimators=50**: This parameter specifies the number of boosting rounds (trees) to build, allowing for a trade-off between performance and complexity.
- **eval_metric='logloss'**: The log-loss metric is optimized during training, a common choice for binary classification problems, minimizing the difference between actual and predicted probabilities.

	Accuracy	Precision(1)	Precision(2)	Recall(1)	Recall(2)	TP	FP	TN	FN
Manual	99.3%	0.99	1.00	1.00	0.97	636	2	2297	18
Info Gain	99.3%	0.99	0.99	1.00	0.98	638	4	2295	16
Correlation	99.3%	0.99	0.99	1.00	0.98	638	4	2295	16

b) **KNN:** The KNeighbors Classifier from the sklearn.neighbors module is an implementation of the k-nearest neighbors (KNN) algorithm, a simple yet effective classification technique based on proximity.

- **n_neighbors=5**: This parameter specifies that the classifier considers the 5 closest training examples (neighbors) to determine the class of a new data point. A majority vote among these neighbors decides the predicted class.
- **Distance Metric**: By default, KNN uses Euclidean distance to measure similarity between points, meaning closer points have more influence on classification.

	Accuracy	Precision(1)	Precision(2)	Recall(1)	Recall(2)	TP	FP	TN	FN
Manual	92.0%	0.94	0.86	0.96	0.76	500	81	2218	154
Info Gain	92.0%	0.92	0.83	0.96	0.71	467	99	2200	187
Correlation	92.4%	0.94	0.87	0.97	0.78	509	78	2221	145

Precision(1), Precision(2), Recall(1) and Recall(2) are precision and recall values for the two lables.
(1)- 0 - False
(2)- 1- True

c) **Decision Tree:** The DecisionTreeClassifier from sklearn.tree is an implementation of a decision tree algorithm, a model that makes classifications by splitting data based on feature values, forming a tree-like structure of decisions. The algorithm recursively splits the dataset into branches based on feature values, aiming to improve homogeneity within each branch. Each split in the tree represents a decision rule, making the model easy to interpret and visualize. This transparency allows insights into how features contribute to the predictions.

	Accuracy	**Precision(1)**	**Precision(2)**	**Recall(1)**	**Recall(2)**	**TP**	**FP**	**TN**	**FN**
Manual	98.4%	0.99	0.96	0.99	0.97	633	26	2273	21
Info Gain	98.7%	0.99	0.97	0.99	0.98	638	23	2276	16
Correlation	98.5%	0.99	0.96	0.99	0.97	635	24	2275	19

3. Best model and subset selection:

Based on the results, the **XGB classifier** turns out to be the best model for purpose of fraudulent transactions detection in the cryptocurrency.

Reasons:

- **Superior Performance**: It maintained the highest accuracy across feature sets.
  High Accuracy: 99.3%
  High Precision: 0.99
  High Recall: 0.97-1.00
  Overall lowest FN values across all datasets.
- **Robustness to Feature Sets**: XGBoost's performance remained stable, indicating adaptability to various selected feature subsets.
- **Balanced Metrics**: Precision and recall were consistently high, making it ideal for datasets where both false positives and false negatives are critical, such as fraudulent transactions detection.

Also, attribute subset based on the correlation with the target variable, i.e, **feat_correlation_subset.csv** turns out to be the optimal subset for this task.

Reasons:

- **Small number of attributes:** This subset contains only **top 20** most relevant features, reducing model complexity while retaining important information. A smaller feature set also speeds up training and inference times, making the model more efficient.
- **Lowest FN value:** This subset minimizes the number of false negatives, meaning that it successfully detects a higher proportion of actual positive cases (fraudulent transactions). Lower FNs are crucial in fraud detection, where missing a fraudulent transaction could have severe consequences.

Precision(1), Precision(2), Recall(1) and Recall(2) are precision and recall values for the two lables.
(1)- 0 - False
(2)- 1- True