



CREDIT CARD FRAUD DETECTION

— IBM NAAN MUDHALVAN



ABSTRACT

The purpose of this project is to detect the fraudulent transactions made by credit cards by the use of machine learning techniques, to stop fraudsters from the unauthorized usage of customers' accounts. The increase of credit card fraud is growing rapidly worldwide, which is the reason actions should be taken to stop fraudsters. Putting a limit for those actions would have a positive impact on the customers as their money would be recovered and retrieved back into their accounts and they won't be charged for items or services that were not purchased by them which is the main goal of the project. Detection of the fraudulent transactions will be made by using three machine learning techniques KNN, SVM and Logistic Regression, those models will be used on a credit card transaction dataset.

How does credit card fraud works ?

Credit card fraud happens when a fraudster gets hold of someone else's credit card details and makes a purchase with it. This is clear fraud, where the goal is to not pay for a good or service and still receive it.

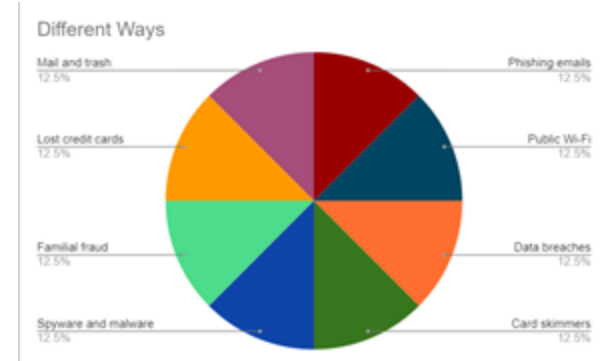
Note that there is also another type of credit card fraud that happens when the cardholder is being dishonest. In that scenario, the payment looks legitimate, but the cardholder has already decided to return the item or ask for a refund.

The latter is called friendly fraud, and it can be challenging to detect. The cardholder may say that the card has been stolen whereas, in fact, they were the one who made the purchase but claim otherwise.

In both scenarios, however, the key ingredients are the same. For credit card fraud to work you need:

- A credit card number (legitimate or stolen).
- A CNP purchase (card not present), for instance at an online store.
- A request for a refund. This will be made by the victim whose stolen card was used in the fraudulent purchase or the legitimate cardholder.

Ideally, however, you will flag credit card fraud before the purchase goes through, ensuring you do not have to deal with the last step.



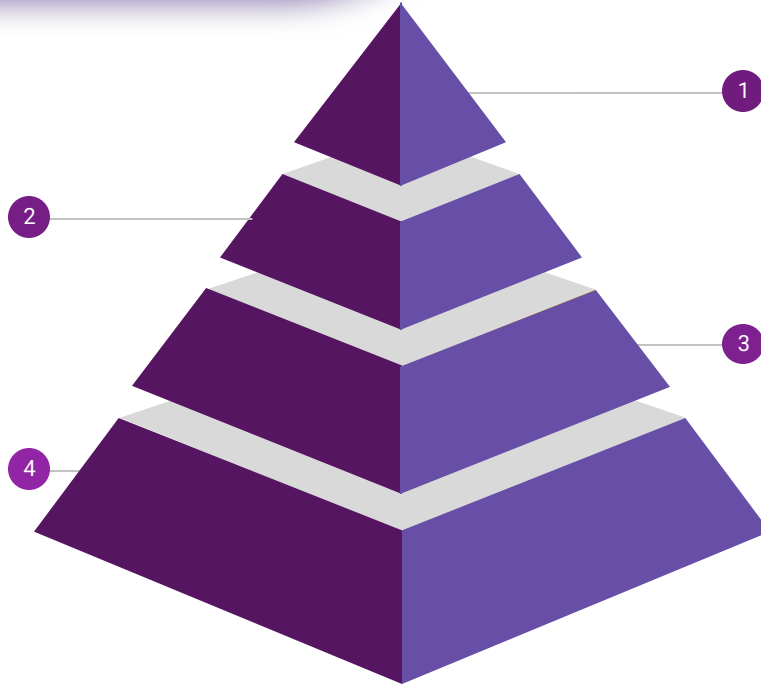
Main challenges

1. Enormous Data is processed every day and the model build must be fast enough to respond to the scam in time.
2. Imbalanced Data i.e most of the transactions (99.8%) are not fraudulent which makes it really hard for detecting the fraudulent ones
3. Data availability as the data is mostly private.
4. Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported.
5. Adaptive techniques used against the model by the scammers.

Technology used

Artificial intelligence and Python
Generative
Adversarial Network (GAN)

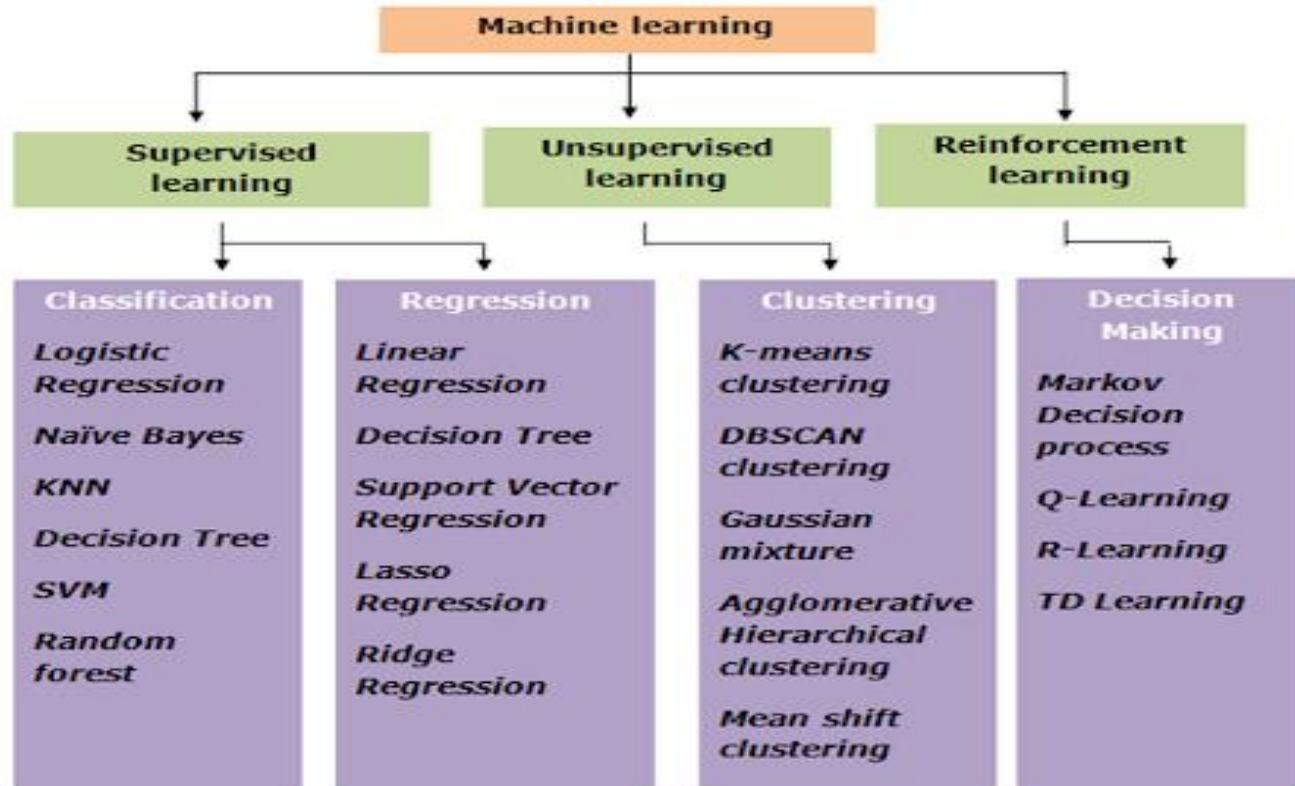
Data science
Data mining to classify, cluster, and segment the data and automatically find associations and rules in the data that may signify interesting patterns, including those related to fraud



Statistical Methodology
statistical parameters such as averages, quantiles, performance metrics, probability distributions,

Machine Learning
Logistic Regression, Random Forest, Naive Bayes and Multilayer Perceptron

Details



System requirement

Software

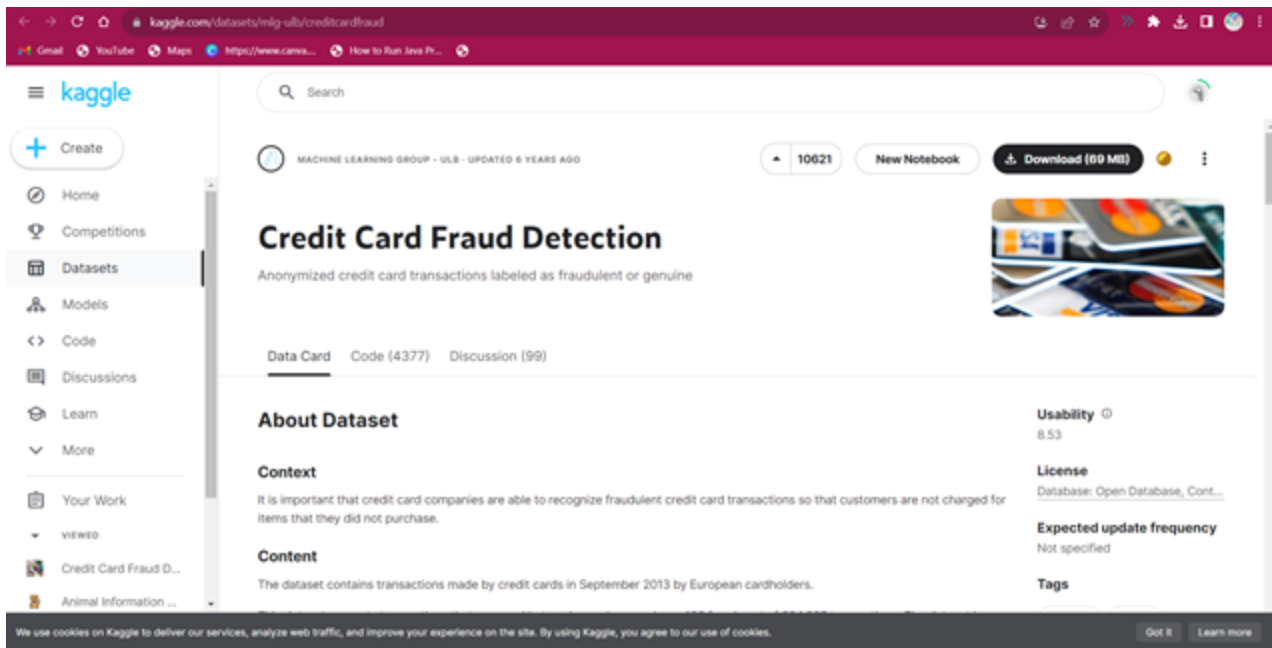
1. Windows Xp, Windows 7(ultimate , enterprise)
2. Python 3.6 and related libraries

Hardware

1. Processor - i3
2. Memory - 1GB RAM
3. Hard Disk - 5 GB

About Dataset

Kaggle Dataset: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>



The screenshot shows the Kaggle website interface for the 'Credit Card Fraud Detection' dataset. The browser address bar displays the URL <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>. The left sidebar contains navigation links: Home, Competitions, Datasets (highlighted), Models, Code, Discussions, Learn, More, Your Work, and a 'VIEWED' section listing 'Credit Card Fraud D...' and 'Animal Information ...'. The main content area features a search bar, a 'Create' button, and a 'MACHINE LEARNING GROUP - ULB - UPDATED 6 YEARS AGO' header. The dataset title 'Credit Card Fraud Detection' is prominently displayed, followed by the description 'Anonymized credit card transactions labeled as fraudulent or genuine'. Below this, there are tabs for 'Data Card' (selected), 'Code (4377)', and 'Discussion (99)'. The 'About Dataset' section includes a 'Context' paragraph stating the importance of recognizing fraudulent transactions and a 'Content' paragraph stating the dataset contains transactions from September 2013. On the right, there are statistics: '10621' rows, a 'New Notebook' button, and a 'Download (69 MB)' button. A 'Usability' score of 8.53 is shown, along with a 'License' section (Database: Open Database, Cont...) and an 'Expected update frequency' of 'Not specified'. A 'Tags' section is also present. At the bottom, a cookie notice states: 'We use cookies on Kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using Kaggle, you agree to our use of cookies.' with 'Got it' and 'Learn more' buttons.

Our Dataset

creditcard (Read-Only) - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23
2	0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.403993	0.251412	-0.01831	0.277838	-0.681
3	0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.0
4	1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969	-0.12136	-2.26186	0.52498	0.247998	0.771679	0.0
5	1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409	1.965775	-1.23262	-0.20804	-0.1083	0.005274	-0.0
6	2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703	-0.03819	0.803487	0.408542	-0.00943	0.798278	-0.0
7	2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813	0.068653	-0.03319	0.084968	-0.20825	-0.55982	-0.0
8	4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821	-0.61199	-0.04558	-0.21963	-0.16772	-0.27071	-0.0
9	7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.20631	-3.80786	0.615375	1.249376	-0.61947	0.791474	1.757964	-1.32387	0.686133	-0.07613	-1.22213	-0.35822	0.324505	-0.15674	1.943465	-1.01545	0.0
10	7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977	0.118765	0.570328	0.052736	-0.07343	-0.26809	-0.0
11	9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098	0.476677	0.451773	0.203711	-0.24691	-0.63375	-0.0
12	10	1.440044	-1.17634	-0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	0.626659	1.199644	-0.67144	-0.51395	-0.09055	0.23093	0.031967	0.253415	0.854344	-0.22137	-0.38723	-0.0093	0.313894	-0.0
13	10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998	0.359985	0.707664	0.125992	0.049924	0.238422	-0.0
14	10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-0.29401	1.333729	0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936	-0.84779	-0.68319	-0.10276	-0.23181	-0.48329	-0.0
15	11	1.069374	0.287722	0.828613	1.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005	-0.9805	-0.82922	-0.1532	-0.03688	0.074412	-0.0
16	12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587	0.778265	2.221868	-1.58212	1.151663	0.222182	-1.0
17	12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731	-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927	-0.41999	0.432535	0.263451	0.499625	1.35365	-0.0
18	12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798	-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24661	-0.00921	-0.59591	-0.57568	-0.11391	-0.02461	0.196002	-0.0
19	13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798	0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871	0.68047	0.025436	-0.04702	-0.1948	-0.67264	-0.0
20	14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.59974	0.160842	1.23309	0.345173	0.91723	0.970117	-0.26657	-0.47913	-0.52661	0.472004	-0.72548	0.075081	-0.40687	-0.25985	-0.5036	0.98446	-0.0
21	15	1.492936	-1.02935	0.454795	-1.43803	-1.55543	-0.72096	-1.08066	-0.05313	-1.97868	1.638076	1.077542	-0.63205	-0.41696	0.052011	-0.04298	-0.16643	0.304241	0.554432	0.05423	-0.38791	-0.17765	-0.17507	-0.0
22	16	0.694885	-0.13633	1.202921	0.834159	-1.19121	1.309190	-0.87859	-0.44529	-0.4462	0.568572	0.191951	0.738329	0.42048	-0.37265	-0.80798	-2.04456	0.515663	0.625847	-1.30041	-0.13833	-0.29558	-0.57196	-0.0
23	17	0.962496	0.328461	-0.17148	2.109204	1.29566	1.696038	0.107712	0.521502	-1.19131	0.724396	1.69033	0.406774	-0.93642	0.983739	0.710911	-0.60223	0.402484	-1.73716	-2.02761	-0.26932	0.143997	0.402492	-0.0
24	18	1.166616	0.50212	-0.0673	2.261569	0.428804	0.089474	0.241147	0.138082	-0.98916	0.922175	0.744786	-0.53138	-2.10535	1.12687	0.003075	0.424425	-0.45448	-0.09887	-0.8166	-0.30717	0.018702	-0.06197	-0.0
25	18	0.247491	0.277666	1.185471	-0.0926	-1.31439	-0.15012	-0.94636	-1.61794	1.544071	-0.92988	-0.5832	0.524933	-0.45338	0.081393	1.555204	-1.39689	0.781131	0.436621	2.177807	-0.23098	1.65018	0.200454	-0.0
26	22	-1.94653	-0.0449	-0.40557	-0.10306	2.941968	2.955053	-0.06306	0.855546	0.049967	0.573743	-0.08126	-0.21575	0.044161	0.033898	1.190718	0.578843	-0.97567	0.044063	0.488603	-0.21672	-0.57953	-0.79923	-0.0
27	22	-2.07429	-0.12148	1.32021	0.410008	0.295198	-0.95954	0.543985	-0.10463	0.475664	0.149451	-0.85657	-0.18052	-0.05523	-0.2798	-0.21167	-0.33332	0.010751	-0.48467	0.505751	-0.38669	-0.40364	-0.2274	-0.0
28	23	1.173285	0.353498	0.283905	1.133563	-0.17258	-0.91605	0.369025	-0.32726	-0.24665	-0.04614	-0.14342	0.97935	1.492285	0.101418	0.761478	-0.01458	-0.51164	-0.32506	-0.39093	0.027878	0.067003	0.227812	-0.0
29	23	1.322707	-0.17404	0.434555	0.576038	-0.83676	-0.83108	-0.2649	-0.22098	-1.07142	0.868559	-0.64151	-0.11132	0.361485	0.171945	0.782167	-1.35587	-0.21694	1.271765	-1.24062	-0.52295	-0.28438	-0.32336	-0.0
30	23	-0.41429	0.905437	1.727453	1.473471	0.007443	-0.20033	0.740228	-0.02295	-0.59339	-0.34619	-0.01214	0.786796	0.635954	-0.08632	0.076804	-1.40592	0.775592	-0.94289	0.543969	0.097308	0.077237	0.457331	-0.0
31	23	1.059387	-0.17532	1.26613	1.18611	-0.786	0.578435	-0.76708	0.401046	0.6995	-0.06474	1.048292	1.005618	-0.542	-0.03991	-0.21868	0.004476	-0.19355	0.042388	-0.27783	-0.17802	0.013676	0.213734	0.0

DATA PREPROCESSING AND FEATURE ENGINEERING

- Data preprocessing:

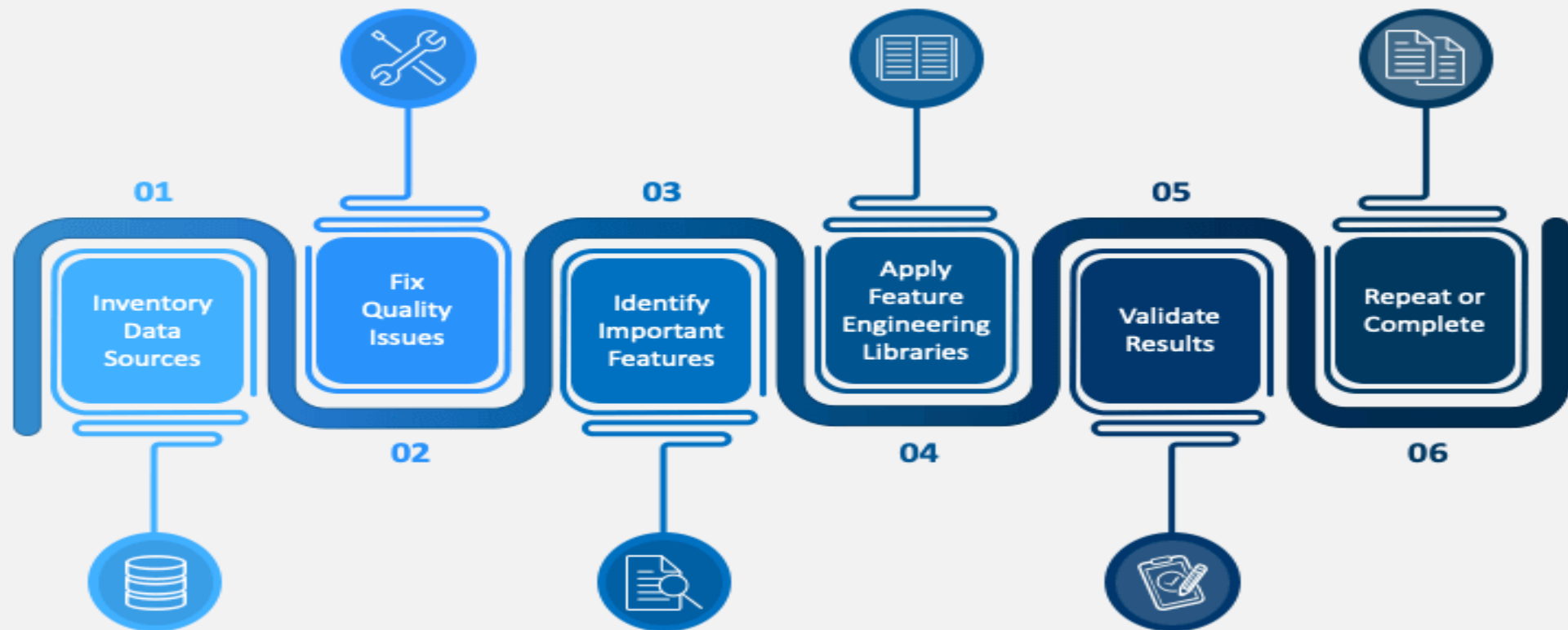
Data preprocessing involves transforming raw data to well-formed data sets so that data mining analytics can be applied. Raw data is often incomplete and has inconsistent formatting. The adequacy or inadequacy of data preparation has a direct correlation with the success of any project that involve [data analytics](#). Preprocessing involves both [data validation](#) and [data imputation](#).

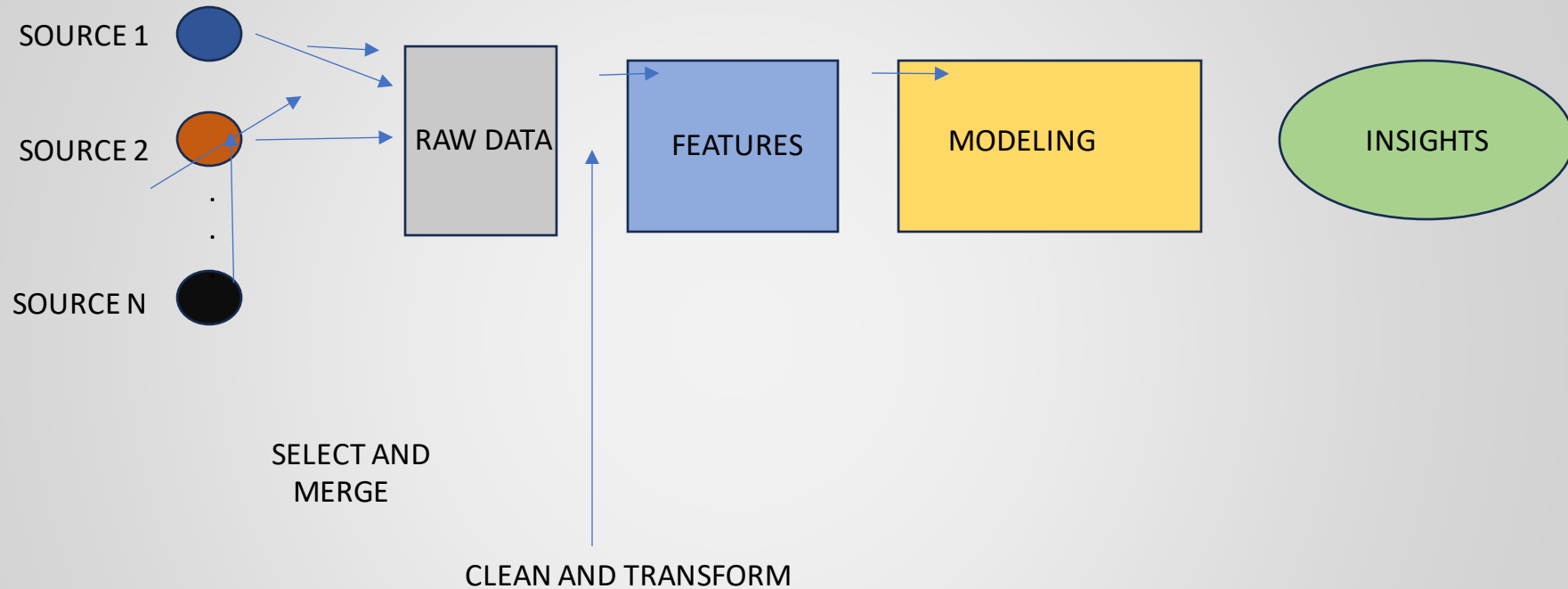
- Feature engineering:

Feature engineering is the pre-processing step of machine learning, which extracts features from raw data. It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data. The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.

DATA PREPROCESSING

Steps for Data Preprocessing





Feature engineering



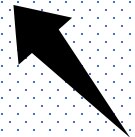
Importance Of Feature Engineering



- ➔ Feature Engineering is a very important step in machine learning.
- ➔ Feature engineering refers to the process of designing artificial features into an algorithm.
- ➔ These artificial features are then used by that algorithm in order to improve its performance,
- ➔ Data scientists spend most of their time with data, and it becomes important to make models accurate.

Importance of data preprocessing

- ➔ By cleaning and formatting the data, we can ensure that the algorithm is only considering relevant information and that it is not being influenced by any irrelevant or incorrect data.
- ➔ it can help to reduce the time and resources required to train the model,
Preprocessing data can also help to prevent overfitting. Overfitting occurs when a model is trained on a dataset that is too specific, and as a result, it performs well on the training data but poorly on new, unseen data.
- ➔ By preprocessing the data and removing irrelevant or redundant information, we can help to reduce the risk of overfitting and improve the model's ability to generalize to new data.



CONCLUSION

The Right Data Preparation Can Lead To Amazing Results Of Good Business Information.

Firms Will Quickly Deal With Data Preparation Challenges With Enhanced Technology And Practises.

Increasing Data Speed, Variety, And Volume Require Companies To Revise The Traditional Sharing, Reporting, And Storage Of Information.

Since Data Is The Basis Of Analytics, The Right Data Will Also Provide Businesses With Crucial Information And Help Them To Actually Respond To Industry Changes

Feature engineering is the development of new data features from raw data.

With this technique, engineers analyze the raw data and potential information in order to extract a new or more valuable set of features.

Feature engineering can be seen as a generalization of mathematical optimization that allows for better analysis.





MODEL SELECTION

Selecting the right model for credit card fraud detection is crucial for building an effective and efficient system. Credit card fraud detection is typically treated as a binary classification problem where the goal is to classify transactions as either "fraudulent" or "legitimate." Here are some steps to help you choose an appropriate model:

Understand the Data	Data Preprocessing	Feature Engineering	Model Evaluation Metrics
Data Splitting	Baseline Models	Advanced Models	Ensemble Methods
Hyperparameter Tuning	Imbalanced Data Handling	Regularization and Robustness	Monitoring and Updates
Model Explainability	Benchmarking	Deployment and Monitoring	Feedback Loop
Regulatory Compliance	Documentation	Robustness Testing	Collaboration with Experts

Fraud detection is a critical application of machine learning, and the choice of algorithms can significantly impact the effectiveness of your system. Below are some machine learning algorithms commonly used in fraud detection:

Logistic Regression:

- Logistic regression is a simple and interpretable algorithm that works well for binary classification problems. It can serve as a baseline model for fraud detection.

Decision Trees:

- Decision trees are used for both classification and regression tasks. They are interpretable and can capture non-linear relationships in the data. However, they are prone to overfitting.

Random Forest:

- Random forests are an ensemble method that consists of multiple decision trees. They offer improved generalization and can handle high-dimensional data. Random forests are less prone to overfitting compared to individual decision trees.

Gradient Boosting Algorithms:

- Algorithms like XGBoost, LightGBM, and CatBoost are widely used for fraud detection. They build an ensemble of decision trees sequentially, with each tree correcting the errors of the previous ones. They are known for their high predictive accuracy.

Support Vector Machines (SVM):

- SVMs work well in high-dimensional spaces and can handle both linear and non-linear classification. They are effective for separating classes, even when the data is not linearly separable.

Neural Networks:

- Deep learning models, such as feedforward neural networks or convolutional neural networks (CNNs), can capture complex patterns in data. They are highly flexible but require large amounts of data and computational resources.

K-Nearest Neighbors (K-NN):

- K-NN is a simple instance-based algorithm that classifies data points based on the majority class of their k-nearest neighbors. It can be effective for fraud detection, especially when the decision boundary is complex.

Isolation Forest:

- Isolation Forest is an anomaly detection algorithm that is suitable for identifying rare and unusual events, making it useful for fraud detection tasks.

One-Class SVM:

- One-Class SVM is another anomaly detection algorithm that can be used when you have a limited amount of labeled fraud data. It aims to separate the target class (fraud) from the rest of the data.

Ensemble Methods:

- Combining multiple models using ensemble techniques like bagging and boosting can enhance fraud detection performance. For example, you can combine decision trees, SVMs, or neural networks into an ensemble.

Autoencoders:

- Autoencoders are neural network architectures used for unsupervised learning and anomaly detection. They can be trained to reconstruct normal data and identify anomalies by detecting reconstruction errors.

Naive Bayes:

- Naive Bayes classifiers are simple probabilistic models that can be used for fraud detection, especially when dealing with categorical data or text-based features.

The choice of algorithm depends on various factors, including the nature of your data, the size of your dataset, the level of class imbalance, computational resources, and the desired level of interpretability. Often, a combination of multiple algorithms or an ensemble approach can yield the best results in fraud detection systems. Additionally, feature engineering, data preprocessing, and model evaluation are also crucial components of building an effective fraud detection system.



Model Training

What is Model Training?

➤ Model training is at the heart of the data science development lifecycle where the data science team works to fit the best weights and biases to an algorithm to minimize the loss function over prediction range. Loss functions define how to optimize the ML algorithms. A data science team may use different types of loss functions depending on the project objectives, the type of data used and the type of algorithm.

➤ When a supervised learning technique is used, model training creates a mathematical representation of the relationship between the data features and a target label. In unsupervised learning, it creates a mathematical representation among the data features themselves.

Importance of Model Training:

➤ Model training is the primary step in machine learning, resulting in a working model that can then be validated, tested and deployed. The model's performance during training will eventually determine how well it will work when it is eventually put into an application for the end-users.

➤ Both the quality of the training data and the choice of the algorithm are central to the model training phase. In most cases, training data is split into two sets for training and then validation and testing.

➤ The selection of the algorithm is primarily determined by the end-use case. However, there are always additional factors that need to be considered, such as algorithm-model complexity, performance, interpretability, computer resource requirements, and speed. Balancing out these various requirements can make selecting algorithms an involved and complicated process.

How To Train a Machine Learning Model:

Split the Dataset

► Your initial training data is a limited resource that needs to be allocated carefully. Some of it can be used to train your model, and some of it can be used to test your model – but you can't use the same data for each step. You can't properly test a model unless you have given it a new data set that it hasn't encountered before. Splitting the training data into two or more sets allows you to train and then validate the model using a single source of data. This allows you to see if the model is overfit, meaning that it performs well with the training data but poorly with the test data.

► A common way of splitting the training data is to use cross-validation. In [10-fold cross-validation](#), for example, the data is split into ten sets, allowing you to train and test the data ten times. To do this:

1. Split the data into ten equal parts or folds.
2. Designate one fold as the hold-out fold.
3. Train the model on the other nine folds.
4. Test the model on the hold-out fold
5. Repeat this process ten times, each time selecting a different fold to be the hold-out fold.

Fit and Tune Models:

► Now that the data is prepared and the model's hyperparameters have been determined, it's time to start training the models. The process is essentially to loop through the different algorithms using each set of hyperparameter values you've decided to explore. To do this:

1. Split the data.
2. Select an algorithm.
3. Tune the hyperparameter values.
4. Train the model.
5. Select another algorithm and repeat steps 3 and 4..

► Next, select another set of hyperparameter values you want to try for the same algorithm, cross-validate it again and calculate the new score. Once you have tried each hyperparameter value, you can repeat these same steps for additional algorithms.

Choose the Best Model:

➤ Now it's time to test the best versions of each algorithm to determine which gives you the best model overall.

1. Make predictions on your test data.
2. Determine the ground truth for your target variable during the training of that model.
3. Determine the performance metrics from your predictions and the ground truth target variable.
4. Run each finalist model with the test data.

➤ Once the testing is done, you can compare their performance to determine which are the better models. The overall winner should have performed well (if not the best) in training as well as in testing. It should also perform well on your other performance metrics (like speed and empirical loss), and – ultimately – it should adequately solve or answer the question posed in your problem statement.



MODEL

EVALUATION

Evaluating the performance of a fraud detection system is critical to ensure its effectiveness in identifying and preventing fraudulent activities while minimizing false positives. Below are common evaluation metrics and techniques used to assess the performance of a fraud detection system:

Confusion Matrix:

- A confusion matrix provides a clear breakdown of model predictions:
 - True Positives (TP): Fraudulent transactions correctly identified as fraud.
 - True Negatives (TN): Legitimate transactions correctly identified as legitimate.
 - False Positives (FP): Legitimate transactions incorrectly classified as fraud (false alarms).
 - False Negatives (FN): Fraudulent transactions incorrectly classified as legitimate (missed frauds).

Precision:

- Precision measures the accuracy of positive predictions (frauds) made by the model. It is calculated as $TP / (TP + FP)$.
- High precision indicates that when the model predicts fraud, it is likely to be correct, but it may miss some actual fraud cases.

Recall (Sensitivity or True Positive Rate):

- Recall measures the ability of the model to identify all actual fraud cases. It is calculated as $TP / (TP + FN)$.
- High recall indicates that the model is good at catching fraud, but it may generate more false alarms.

F1-Score:

- The F1-score is the harmonic mean of precision and recall and provides a balanced measure of a model's performance. It is calculated as $2 * (Precision * Recall) / (Precision + Recall)$.

Area Under the ROC Curve (AUC-ROC):

- ROC (Receiver Operating Characteristic) curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold values.
- AUC-ROC measures the model's ability to distinguish between classes, with a higher AUC indicating better discrimination.

Area Under the Precision-Recall Curve (AUC-PR):

- The precision-recall curve plots precision against recall at various threshold values.
- AUC-PR summarizes the trade-off between precision and recall, especially when dealing with imbalanced datasets.

Accuracy:

- While not the primary metric for fraud detection (due to class imbalance), accuracy is still useful as a general measure of model performance. It is calculated as $(TP + TN) / (TP + TN + FP + FN)$.

Specificity:

- Specificity measures the model's ability to correctly identify legitimate transactions. It is calculated as $TN / (TN + FP)$.

False Positive Rate (FPR):

- FPR measures the rate of legitimate transactions incorrectly classified as fraud. It is calculated as $1 - \text{Specificity}$.

Cost-Based Metrics:

- In some cases, you may want to consider the financial impact of false positives and false negatives. Cost-sensitive metrics can help optimize the model based on these considerations.

Threshold Selection:

- Depending on business requirements, adjust the classification threshold to balance precision and recall or minimize false positives.

Cross-Validation:

- Use k-fold cross-validation to assess the model's performance across different subsets of the data and check for overfitting.

Concept Drift Monitoring:

- Continuously monitor the model's performance over time to detect concept drift, which occurs when the data distribution changes. Update the model as needed.

Anomaly Detection Techniques:

- Consider anomaly detection techniques like isolation forest, one-class SVM, or autoencoders to identify previously unseen fraud patterns.

Model Explainability:

- Use techniques like SHAP values or feature importance to gain insights into why the model makes certain predictions. Explainability is crucial for regulatory compliance and trust.

Business Impact Analysis:

- Assess the real-world impact of your fraud detection system, including how well it reduces fraud losses and false positives while improving operational efficiency.

It's important to choose the evaluation metrics that align with your specific fraud detection goals and business priorities. Additionally, regularly update and refine your fraud detection system to adapt to evolving fraud tactics and changing data distributions.

Thank you!