https://www.kaggle.com/c/comp90051-2021-semester-2-project-1/data?select=train.csv

Approaches for imbalanced data classification

- https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18
- https://towardsdatascience.com/4-unique-approaches-to-manage-imbalance-classification-scenario-7c5b92637b9c
- https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
- https://towardsdatascience.com/guide-to-classification-on-imbalanced-datasets-d6653aa5fa23
- https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/
-

- Performance metric:
  - Accuracy is not the best metric for imbalance data may mislead of the prediction performace)
  - Confusion Matrix: a table showing correct predictions and types of incorrect predictions.
    - https://towardsdatascience.com/accuracy-visualisation-supervised-machine-learning-classification-algorithms-af13d18fcc6c
  - Precision: the number of true positives divided by all positive predictions. Precision is also called Positive Predictive Value. It is a measure of a classifier's exactness. Low precision indicates a high number of false positives.
  - Recall: the number of true positives divided by the number of positive values in the test data. The recall is also called Sensitivity or the True Positive Rate. It is a measure of a classifier's completeness. Low recall indicates a high number of false negatives.
  - F1 Score: the weighted average of precision and recall. (may mislead of the prediction performace)
  - Kappa (or Cohen's kappa): Classification accuracy normalized by the imbalance of the classes in the data.
  - ROC Curves: Like precision and recall, accuracy is divided into sensitivity and specificity and models can be chosen based on the balance thresholds of these values.
    - https://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/
  - Summary:
    - AUROC and accuracy is not suitable metrics for imbalance datasets as they can be misleading by the imbalance data
    - Usually recommend using confusion matrix, recall, precision & f1 together

- Change the algorithm:
  - Decision trees frequently perform well on imbalanced data. They work by learning a hierarchy of if/else questions and this can force both classes to be addressed.
    - That being said, decision trees often perform well on imbalanced datasets. The splitting rules that look at the class variable used in the creation of the trees, can force both classes to be addressed.

        - If in doubt, try a few popular decision tree algorithms like C4.5, C5.0, CART, and Random Forest.
        - For some example R code using decision trees, see my post titled "Non-Linear Classification in R with Decision Trees".
        - For an example of using CART in Python and scikit-learn, see my post titled "Get Your Hands Dirty With Scikit-Learn Now".
    - Random forest
        - Random Forrest Classifier has a parameter "class_weight" to specify the weights of each class in case of an imbalanced dataset.
        - `RandomForestClassifier`(w/ unbalanced data w/ class_weight) is prefered
            - Cost sensitive for extreme imbalance
        - `RandomForestClassifier`(w/ balanced data w/out class_weight)
        - `RandomForestClassifier`(w/ SMOTE w/out class_weight)
        - `BalancedRandomForestClassifier`(w/ unbalanced data) is prefered
            - sampling
    - Penalized model (cost sensitive training)
        - Penalized classification imposes an additional cost on the model for making classification mistakes on the minority class during training. These penalties can bias the model to pay more attention to the **minority class**.
        - Often the handling of class penalties or weights are specialized to the learning algorithm. There are penalized versions of algorithms such as **penalized-SVM and penalized-LDA.**
        - It is also possible to have **generic frameworks** for penalized models. For example, Weka has a CostSensitiveClassifier that can wrap any classifier and apply a custom penalty matrix for miss classification.
        - Using penalization is desirable if you are **locked into a specific algorithm and are unable to resample or you're getting poor results**. It provides yet another way to **"balance" the classes**. Setting up the penalty matrix can be complex. You will very likely have to try a variety of penalty schemes and see what works best for your problem.

- Sampling methods: act on the data
    - oversample minority class => have an equal ratio of data points
        - adding more copies(generate new data) of the minority class.
        - generate points that are close in dataspace proximity to existing samples or are 'between' two samples/ randomly replicate samples from the minority class
        - Oversampling can be a good choice when you don't have a ton of data to work with.
        - Always split into test and train sets BEFORE trying oversampling techniques!
    - Undersample majority class => have an equal ratio of data points
        - Remove from majority class
        - Undersampling can be a good choice when you have a ton of data -think millions of rows. But a drawback is that we are removing information that may be valuable. This could lead to underfitting and poor generalization to the test set.
        - Generate synthetic sample
    - Summary for above methods:
        - Add false data -> risk of overfitting

        - contribute additional noise to our model
    - Similar to upsampling(SMOTE)
        - will use imblearn's SMOTE or Synthetic Minority Oversampling Technique
        - SMOTE: SMOTE uses the nearest neighbours algorithm based on their local density and their borders with the other class to generate new and synthetic data we can use for training our model.
        - Not only does it perform oversampling, but can subsequently use cleaning techniques (undersampling, more on this shortly) to remove redundancy in the end.
        - Again, it's important to generate the new samples only in the training set to ensure our model generalizes well to unseen data.
        - https://blog.csdn.net/u010654299/article/details/103980964
    - Summary for SMOTE:
        - generalizes the decision region for the minority class. As a result, larger and less specific regions are learned, thus, paying attention to minority class samples without causing overfitting.
        - Overgeneralization. SMOTE's procedure can be dangerous since it blindly generalizes the minority area without regard to the majority class. This strategy is particularly problematic in the case of highly skewed class distributions since, in such cases, the minority class is very sparse with respect to the majority class, thus resulting in a greater chance of class mixture.
        - Inflexibility. The number of synthetic samples generated by SMOTE is fixed in advance, thus not allowing for any flexibility in the re-balancing rate.
        - Another potential issue is that SMOTE might introduce the artificial minority class examples too deeply in the majority class space. This drawback can be resolved by hybridization: combining SMOTE with undersampling algorithms. One of the most famous of these is Tomek Links. Tomek Links are pairs of instances of opposite classes who are their own nearest neighbors. In other words, they are pairs of opposing instances that are very close together. Tomek's algorithm looks for such pairs and removes the majority instance of the pair. The idea is to clarify the border between the minority and majority classes, making the minority region(s) more distinct.
        - Other more nuanced versions of SMOTE include Borderline SMOTE, SVMSMOTE, and KMeansSMOTE, and more nuanced versions of the undersampling techniques applied in concert with SMOTE are Condensed Nearest Neighbor (CNN), Repeated Edited Nearest Neighbor, and Instance Hardness Threshold.

- Cost-sensitive methods
    - Upweighting. Upweighting is analogous to over-sampling and works by increasing the weight of one of the classes keeping the weight of the other class at one.
    - Down-weighting. Down-weighting is analogous to under-sampling and works by decreasing the weight of one of the classes keeping the weight of the other class at one.
    -
- Random forest and SMOTE could be the best option for imbalanced data
- Try different perspective
    - Anomaly detection is the detection of **rare events**. This might be a machine malfunction indicated through its vibrations or malicious activity by a program indicated by it's sequence of system calls. The events are rare and when compared to normal operation.

- This shift in thinking considers **the minor class as the outliers class** which might help you think of new ways to separate and classify samples.
- Change detection is similar to anomaly detection except rather than looking for an anomaly it is looking for **a change or difference**. This might be a **change in behaviour** of a user as observed by usage patterns or bank transactions.
- Both of these shifts take a mor**e real-time stance** to the classification problem that might give you some new ways of thinking about your problem and maybe some more techniques to try.
-

Problem defined:

https://machinelearningmastery.com/what-is-imbalanced-classification/
https://towardsdatascience.com/programming-fairness-in-algorithms-4943a13dd9f8

3d joint
https://paperswithcode.com/task/3d-human-action-recognition
https://github.com/microsoft/tensorwatch
https://paperswithcode.com/task/skeleton-based-action-recognition
https://hal.archives-ouvertes.fr/hal-01823804/document
https://github.com/jinwchoi/awesome-action-recognition
https://www.sciencedirect.com/science/article/pii/S0167865521000751
file:///Users/apple/Dropbox/2021_SM2/electronics-10-01118-v3.pdf
https://www.ijcai.org/Proceedings/16/Papers/470.pdf

- Unequal distribution of classes (Classes are biased or skewed)
- Ml algorithm for classification is designed based on the assumption of an equal number of examples for each class
- Poor predictive performance on the minority class
  - standard classification algorithms do not work as they try to minimize the error rate rather than focus on the minority class, giving bias classification.
- More sensitive to classification errors for minority
- We are facing the multiclassification problem
- Type of imbalance: Between-class
- Class overlap?
  - the class more represented in overlap regions tends to be better classified by methods based on global learning (on the full dataset). This is because the algorithm is able to get a more informed picture of the data distribution of the majority class.

Logistic regression for imbalanced data classification ❌

https://machinelearningmastery.com/multinomial-logistic-regression-with-python/

https://machinelearningmastery.com/cost-sensitive-logistic-regression/

- Logistic regression does not support imbalanced classification directly.
- Instead, the training algorithm used to fit the logistic regression model must be modified to take the skewed distribution into account.
- This can be achieved by specifying a class weighting configuration that is used to influence the amount that logistic regression coefficients are updated during training.
- Logistic regression, by default, is limited to two-class classification problems. By default, logistic regression cannot be used for classification tasks that have more than two class labels, so-called multi-class classification.
- Instead, it requires modification to support multi-class classification problems.
- Some extensions like one-vs-rest can allow logistic regression to be used for multi-class classification problems, although they require that the classification problem first be transformed into multiple binary classification problems.
- An alternate approach involves changing the logistic regression model to support the prediction of multiple class labels directly. Specifically, to predict the probability that an input example belongs to each known class label.

Random forest

https://medium.com/sfu-cspmp/surviving-in-a-random-forest-with-imbalanced-datasets-b98b963d52eb
https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf
https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/

- Random forest algorithm is one of the most popular and potent supervised machine learning algorithms capable of performing both classification and regression tasks. This algorithm creates a forest with several decision trees. The more trees in the forest, the more robust the prediction is and hence higher accuracy to model multiple decision trees.
- To create the forest, one will use the same method of constructing the decision tree with the information gain or Gini index approach amongst other algorithms.
- In random forests, we grow multiple trees instead of a single tree in the model to classify a new object. Based on the attributes, **each tree gives a classification, and the forest chooses the class with the most votes as the classifier**. In the case of regression, it takes the **average** of the outputs by different trees.
- Why rf is ideal?
    - Firstly, the ability to incorporate class weights into the random forest classifier makes it cost-sensitive; hence it penalizes misclassifying the minority class.
    - Secondly, it combines the sampling technique and ensemble learning, therefore, downsampling the majority class and growing trees on a more balanced dataset.

    - The algorithm is a strong modelling technique and is much more sturdy than a single decision tree. The aggregation of several trees **limits the possibility of overfitting and miscalculations due to bias and**, in return, capitulates handy results.
    - Random forest classifier handles the missing values and maintains accuracy for missing data when a large proportion of the data is missing. **It has the power to control large data sets with higher dimensionality.**
- stratified sampling
    - ensuring splitting the data randomly and keeping the same imbalanced class distribution for each subset. The modified version of K-Fold i.e. stratified K-Fold Cross Validation necessitates the matching class distribution with the complete training dataset in each split.
- Models analysis:
    - Standard random forest
        - Each decision tree in the forest is created from randomly selected bootstrap samples, which are aggregated to make a prediction called 'Bagging.' This step allows the forests' diversity, resulting in substantial performance improvement in its accuracy
        - the random forest classifier is also designed to minimize the overall error rate. Thus, we cannot expect a decent result in predicting minority class values with an imbalanced dataset.
        - SRF has its restrictions with imbalanced classes because it uses a bootstrap sample of the training set to form each tree. In imbalance learning, the likelihood of bootstrap samples containing few or none of the minority classes increases notably, resulting in a model being a lousy predictor of the minority class.
    - Balanced random forest
        - it is crucial to make class priors equal, either by downsampling or oversampling. Hence, BRF does this by iteratively drawing a bootstrap sample with equal proportions of data points from both the minority and the majority class
        - Unfortunately, BRF was not much beneficial in improving the class imbalance issue with this dataset in particular.
    - Use SRF on the SMOTE data

Books

- Chapter 16: Remedies for Severe Class Imbalance, Applied Predictive Modeling, 2013.
- Imbalanced Learning: Foundations, Algorithms, and Applications, 2013.
- Learning from Imbalanced Data Sets, 2018.

XGboost:

https://machinelearningmastery.com/xgboost-for-imbalanced-classification/
http://5.9.10.113/67868420/xgboost-for-multiclassification-and-imbalanced-data

Lightgbm:

https://www.kaggle.com/pranav84/lightgbm-fixing-unbalanced-data-lb-0-9680
https://towardsdatascience.com/lightgbm-with-the-focal-loss-for-imbalanced-datasets-9836a9ae00ca

https://medium.com/@cmukesh8688/how-to-handle-imbalanced-classification-problems-4a96f42ae4c4
https://www.kaggle.com/shahules/tackling-class-imbalance
https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/
https://neptune.ai/blog/lightgbm-parameters-guide

特征提取:

https://www.sciencedirect.com/science/article/pii/S0167865521000751
忽略时序性

16*60

LSTM:
https://www.kaggle.com/sanjay11100/lstm-sentiment-analysis-data-imbalance-keras
https://www.hindawi.com/journals/complexity/2020/9084704/
https://datascience.stackexchange.com/questions/17219/how-to-do-imbalanced-classification-in-deep-learning-tensorflow-rnn
https://stats.stackexchange.com/questions/342170/how-to-train-an-lstm-when-the-sequence-has-imbalanced-classes

SVM:

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
- https://scikit-learn.org/stable/modules/svm.html#
- https://xgboost.readthedocs.io/en/latest/python/python_api.html
- https://neptune.ai/blog/lightgbm-parameters-guide
- https://github.com/derek20F/Unimelb-COMP90051-Project1/blob/master/Code/Alternative%20-%20RFC_new.ipynb
- https://github.com/Bachfischer/COMP90051-StatML-Assignment-1/tree/master/src/03_biggraph
- https://github.com/Bachfischer/COMP90051-StatML-Assignment-1/tree/master/src
- https://github.com/MitPandya/Human-Action-Recognition-and-Video-Classification-using-SVM-and-Deep-CNN-
- https://github.com/MitPandya/Human-Action-Recognition-and-Video-Classification-using-SVM-and-Deep-CNN-/blob/master/svm.py
- https://machinelearningmastery.com/cost-sensitive-svm-for-imbalanced-classification/
- https://towardsdatascience.com/support-vector-machines-imbalanced-data-feb3ecffbb0e
- https://www.sciencedirect.com/science/article/pii/S1877050918322269

Smoteboost:

https://www-users.cse.umn.edu/~lazar027/papers/pkdd03.pdf

https://www.sciencedirect.com/science/article/pii/S1877050918322269

http://dsdeepdive.blogspot.com/2015/12/classifier-boosting-with-python.html

https://link.springer.com/article/10.1007/s00500-014-1407-5

http://www.nwu-ipmi.cn/UploadFile/ReportFile/%E5%9F%BA%E4%BA%8ESMOTEBoost%E7%9A%84%E9%9D%9E%E5%9D%87%E8%A1%A1%E6%95%B0%E6%8D%AE%E9%9B%86SVM%E5%88%86%E7%B1%BB%E5%99%A8_%E6%9D%8E%E6%AD%A3%E6%AC%A3.pdf

https://github.com/dialnd/imbalanced-algorithms

https://github.com/dialnd/imbalanced-algorithms/blob/master/smote.py

https://medium.com/urbint-engineering/using-smoteboost-and-rusboost-to-deal-with-class-imbalance-c18f8bf5b805

https://www.kaggle.com/jrw2200/adaboost-smoteboost-and-rusboost/data

https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74