**Class imbalance problem**

One of the main issues encountered when building the model was class imbalance. Our dataset consisted of 9388 training samples belonging to 49 different classes, with the highest and lowest number of samples in a class being equal to 590 and 24 respectively. This is because certain actions are performed less than others, so naturally there is less data available for certain classes. Thus, using conventional machine learning algorithms would lead to inaccurate and biased classification.

Class imbalance occurs when the number of observations belonging to one class is significantly lower than those belonging to other classes.

The goal is to improve then identification of the rare minority class as opposed to achieving a higher overall accuracy. Define rare event as an event predicted to belong in the minority class with event rate less than 5%.

Approaches to handling imbalanced data – A data level approach

Generally, data level approaches are resampling techniques. Resampling techniques aim to balance classes in the pre-processing stage (i.e. in the training data) before using the data as input. The main goal is to obtain approximately the same number of instances for the classes.

* Random under sampling
  + Randomly eliminate majority class examples
  + i.e. take a certain % of samples (no replacement) from the majority class and combine them with the minority class – treat this as the dataset to build a model
  + Advantage:
    - Help improve run time
    - Reduces the number of training data samples for a large dataset
  + Disadvantage:
    - Discards lots of information that could potentially be useful
    - May be a biased sample since it is random
    - May not accurately represent the population (which can lead to inaccurate results with the actual test data set)
* Random over sampling
  + Increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample
  + Eg. Replicating minority class samples/observations and adding them to the majority class
  + Advantages:
    - No information loss
    - Outperforms undersampling
  + Disadvantages:
    - Increases the likelihood of overfitting
* Cluster based over sampling
  + Apply k means clustering algorithm (independently) to minority and majority class samples – identify clusters in the dataset
  + Then oversample each cluster such that all clusters of the same class have an equal number of instances and all classes have the same size
  + Eg.
    - Total observations = 1000
    - Minority = 20
    - Majority = 980
    - Event rate = 20/1000 = 2%
    - Apply k means
    - Minority: cluster 1: 8 observations, cluster 2: 12 observations
    - Majority: 150,120,230,200,150,130
    - Oversample each cluster
    - Minority: 250,250
    - Majority: 170,170,170,170,170,170
    - Event rate = (250\*2)/(170\*6 + 250\*2) = 33%
  + Advantages
    - Overcomes the challenge between class imbalance (number of examples representing positive class differs from the number of examples representing a negative class)
    - Overcomes challenges within class imbalance, where a class is composed of different sub clusters and each sub cluster does not contain the same number of examples
  + Disadvantage
    - Overfitting possibility
* Informed over sampling (synthetic minority oversampling technique - SMOTE)
  + Avoids overfitting which occurs when exact replicas of minority instances are added to the main dataset
  + Subset of data is a taken from the minority class as an example and then a new synthetic similar instance is created
  + These synthetic instances are added to the dataset
  + The new dataset (original + synthetic) is used as a sample to train the model
  + i.e. take a sample of 15 instances from the minority class (out of 20)
    - Generate similar synthetic instances i.e. 20 times
    - So then we will have 300 (15\*20) synthetic data which we will use as the minority class
    - Event rate = 300/(300 + 980) = 23.4%
  + Advantage:
    - mitigates the problem of overfitting caused by random oversampling as synthetic examples are generated
    - We are not replicated samples from the minority class
    - No loss of information
  + Disadvantages
    - SMOTE does not take into consider neighboring examples from other classes – can result in an increase in overlapping of classes (additional noise)
    - Poor for high dimensional data – i.e. lots of features
* Modified synthetic minority oversampling technique (MSMOTE)
  + Classifies the sample of minority classes into 3 distinct groups called
    - security/safe samples – those that can improve classifier performance
    - border samples – ones that are hard to classify into the other two
    - latent noise samples – reduce the performance of the classifier
  + Calculate the distance among samples of the minority class and samples of the training data
  + For safe samples, algorithm randomly selects a data point from the kNN
  + For border, it selects the nearest neighbor
  + For noise, it ignores it

Handling imbalance data – modifying existing algorithms

* The goal is to modify classification algorithms to make them appropriate for imbalanced datasets
* Called ensemble methodology
  + Construct several two stage classifiers from the original data and then aggregate their predictions
* Bagging based techniques
  + “Bootstrap Aggregating”
  + Generate n different bootstrap training samples with replacement and then training the algorithm on each bootstrapped sample separately and then aggregating the predictions at the end
  + Used for reducing overfitting
  + Bagging difference to boosting is that it allows replacement in the bootstrapped sample
  + Choose a number of bootstrap samples from the population with replacement
  + Fit a machine learning algorithm (decision tree, log reg) to get a classifier
  + Aggregate the classifiers produced for a compound classifier
  + Advantages:
    - Improves stability and accuracy of ML algorithms
    - Reduces variance
    - Overcomes overfitting
    - Improved misclassification rate of the bagged classifier
    - Can outperform boosting in noisy data environments
  + Disadvantages:
    - Bagging only works if the base classifiers are not bad to begin with – if they are bad then it will further degrade performance
* Boosting based techniques
  + Combines weak learners to create a strong learner that can make accurate predictions
  + The base learner/classifier learners where the prediction accuracy is only slightly better than average
  + Classifier algorithm is weak when small changes in the data induces big changes in the classification model
  + In the next iteration, the new classifier focuses on/places more weight to these cases which were incorrectly classified in the last round
* Adaptive boosting (ADA Boost)
  + Creates a highly accurate prediction rule by combining many weak and inaccurate rules
  + Each classifier is serially trained with the goal of correctly classifying examples in every round that were incorrectly classified in the previous round
  + After each round, it gives more focus to examples that are harder to classify
  + Quantity of focus is measured by a weight (initially equal for all instances)
  + After each iteration, the weights of misclassified instances are increased anf the weights of correctly classified instances are decreased
  + Assumption of this algorithm: each of the weak hypothesis has an accuracy slightly better than random guessing

Diagram

Description automatically generated

* + Eg. In a data set containing 1000 observations out of which 20 observations are the minority, equal weights are assigned to all observations and the base classifier accurately classifies 400 observations
  + Now, weight of each of the 600 misclassified observations is increased to w2
  + Weight of each of the correctly classified observations is decreased to w3
  + Updated weighted observations are fed to the weak classifier to improve its performance
  + Continues until the misclassification rate significantly decreases 🡪 a stronger classifier
  + Requires the users to specify a set of weak learners OR randomly generates the weak learners before the actual learning step
  + Advantage:
    - simple to implement
    - good generalization i.e. good for any problem that is not prone to overfitting
  + Disadvantage
    - Sensitive to noisy data
* Gradient tree boosting
  + Many models trained sequentially
  + Decision trees are used as weak learners in gradient boosting
  + Builds the first learner on the training dataset to predict the samples
  + Calculates the loss (difference between real value and output of the first learner)
  + And then uses this loss to build an improved learner in the second stage
  + At every step, the residual loss function is calculated using the gradient descent method
  + The new residual becomes a target variable for the subsequent iteration

Performance metric:

For imbalanced data, it may be better to use confusion matrix, precision, recall, F1:Score, area under ROC Curve rather than accuracy to measure the performance of the classifier

**Models**

Sequential modeling

* Machine learning models that input or output sequences of data
* Recurrent neural networks is a popular algorithm used in sequence models
* Recurrent Neural Networks (RNN)
  + Main advantage over standard neural networks: features are not shared in standard neural networks
    - RNN can remember its previous inputs by standard neural networks are not capable of remembering
    - RNN Loss Function
* Text, schematic

  Description automatically generated
  + One-to-many, many-to-one, many-to-many
  + In our project, it is one to many (take one image or “frame” and output 1 of 49 class labels)
* Long Short Term Memory (LTSM)
  + Method used to overcome the vanishing gradient problem in RNNs
    - Because traditional RNNs are not good at capturing long range dependencies (vanishing gradient problem)
    - When training very deep networks, gradients/derivatives decrease exponentially as it propagates down the layers
    - Gradients are used to update the weights of neural networks, so if the gradients vanish then the weights will not be updated
  + In LSTM, in addition to the hidden state of RNN, a cell state is also passed to the next time step, allowing LSTM to capture long range dependencies
    - Means it can have memory about previous inputs for extended time durations
  + 3 gates: forget, input, output
* Diagram, schematic

  Description automatically generated

Research papers:

Sequential Deep Learning for Human Action Recognition

M Baccouche, F Mamalet

* A fully automated deep model that learns to classify human actions without using any prior knowledge
* Note that the data set used are images
* Two step nueral based model
  + Convolutional neural network extension to the 3d case – will automatically learn spatio temporal features
  + Use these learned features to train a recurrent neural network model in order to classify the entire sequence
* Deep learning of spatio temporal features (extension of convolutional nueral network to 3d)

Ensemble models

* Combine the decisions from multiple models to improve the overall performance

Ada Boost (see class imbalance section)

Random Forest

**Classifier Evaluation**

<https://www.svds.com/the-basics-of-classifier-evaluation-part-1/>

* Standard classifier algorithms have a bias towards classes which have a number of instances i.e. the majority class data
  + Minority class features usually treated as noise
* Generally, classifiers are measured by a confusion matrix that contains information about the actual and predicted class
* Table

  Description automatically generated

Accuracy of a model = (TP+TN)/(TP+FN+FP+TN)

**Human Action Recognition**

Possible report structure: we try to use standard classifier like a decision tree, or maybe use a decision tree as the baseline? Then compare the results of our model. We can probably use ADA boost? Or we could try oversampling and using a decision tree classifier.