MACHINE LEARNING- WORKSHEET 1

1. What is the advantage of hierarchical clustering over K-means clustering?

Ans- B) In hierarchical clustering you don’t need to assign number of clusters in beginning

2. Which of the following hyper parameter(s), when increased may cause random forest to over fit the data?

Ans- A) max_depth

3. Which of the following is the least preferable resampling method in handling imbalance datasets?

Ans- B) RandomOverSampler

4. Which of the following statements is/are true about “Type-1” and “Type-2” errors?

1. Type1 is known as false positive and Type2 is known as false negative.

2. Type1 is known as false negative and Type2 is known as false positive.

3. Type1 error occurs when we reject a null hypothesis when it is actually true.

Ans- C) 1 and 3

5. Arrange the steps of k-means algorithm in the order in which they occur:

1. Randomly selecting the cluster centroids

2. Updating the cluster centroids iteratively

3. Assigning the cluster points to their nearest center

Ans- D) 1-3-2

6. Which of the following algorithms is not advisable to use when you have limited CPU resources and time, and when the data set is relatively large?

Ans- B) Support Vector Machines

7. What is the main difference between CART (Classification and Regression Trees) and CHAID (Chi Square Automatic Interaction Detection) Trees?

Ans- C) CART can only create binary trees (a maximum of two children for a node), and CHAID can create multiway trees (more than two children for a node)

8. In Ridge and Lasso regularization if you take a large value of regularization constant(lambda), which of the following things may occur?

Ans- A) Ridge will lead to some of the coefficients to be very close to 0
D) Lasso will cause some of the coefficients to become 0.

9. Which of the following methods can be used to treat two multi-collinear features?

Ans- B) remove only one of the features
C) Use ridge regularization D) use Lasso regularization

10. After using linear regression, we find that the bias is very low, while the variance is very high. What are the possible reasons for this?

Ans- A) Overfitting

11. In which situation One-hot encoding must be avoided? Which encoding technique can be used in such a case?

Ans- One Hot Encoding is a common way of pre-processing categorical features that creates a new binary feature for each possible category(Thus increasing memory space) and assigns a value of 1 to the feature of each sample that corresponds to its original category (Obtained through Label Encoder).

Thus, One-hot Encoding must be avoided when:

1. **The categorical feature is ordinal (like Jr. kg, Sr. kg, Primary school, high school)**
2. **The number of categories is quite large as one-hot encoding can lead to high memory consumption**

And in such situations, we should stick to Label Encoder.

12. In case of data imbalance problem in classification, what techniques can be used to balance the dataset? Explain them briefly.

Ans- The various techniques one can use to handle imbalanced datasets are as follows:

1. **Random Undersampling and Oversampling:**
   A widely adopted and perhaps the most straightforward method for dealing with highly imbalanced datasets is called resampling. It consists of removing samples from the majority class **(under-sampling)** and/or adding more examples from the minority class **(over-sampling).**

   a. **Undersampling** can be a good choice when you have a ton of data -think millions of rows. But a drawback to undersampling is that we are removing information that may be valuable.
   Syntax- class_0_under = class_0.sample(class_count_1)
   test_under = pd.concat([class_0_under, class_1], axis=0)

   b. **Oversampling** can be defined as adding more copies to the minority class. Oversampling can be a good choice when you don’t have a ton of data to work with. A con to consider when overrsampling is that it can cause overfitting and poor generalization to your test set.

   Syntax- class_1_over = class_1.sample(class_count_0, replace=True)
   test_over = pd.concat([class_1_over, class_0], axis=0)

2. **Undersampling and Oversampling using imbalanced-learn:**
imbalanced-learn(imblearn) is a Python Package to tackle the curse of imbalanced datasets.

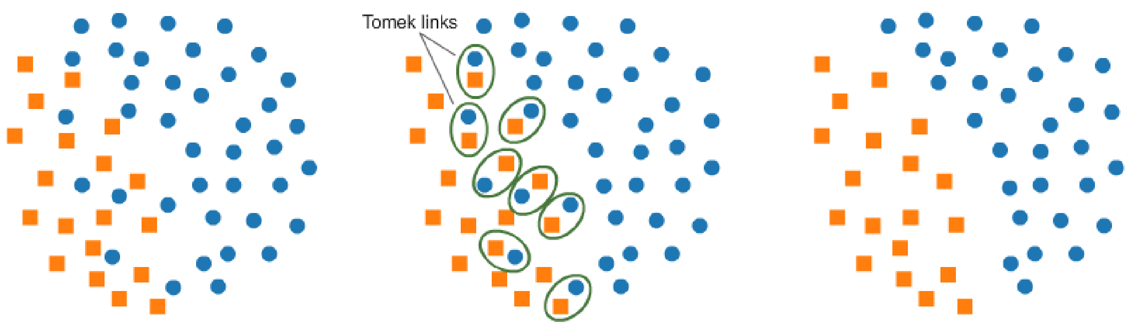
a. **Random under-sampling with imblearn**

Syntax-from imblearn.under_sampling import RandomUnderSampler
rus=RandomUnderSampler(random_state=42,replacement=True)#fit predictor & target
x_rus, y_rus = rus.fit_resample(x, y)

b. **Random over-sampling with imblearn**

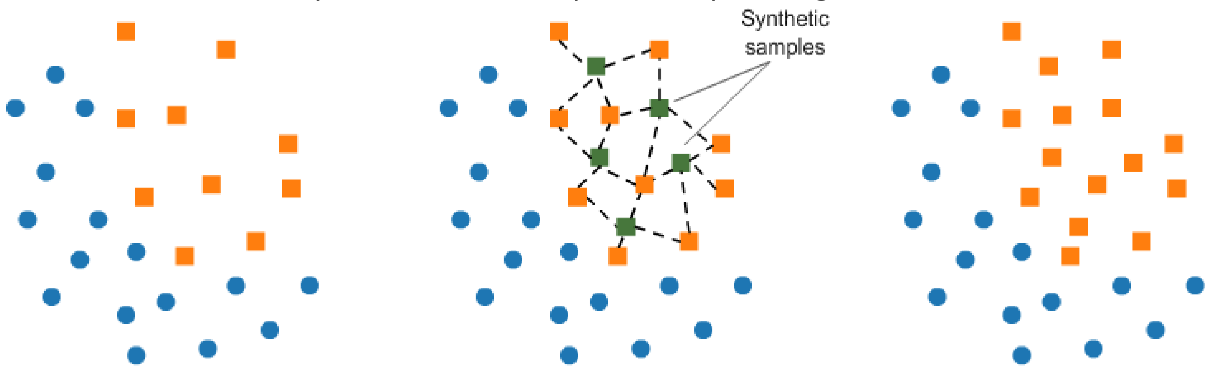
Syntax-from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=42) # fit predictor and target variable
x_ros, y_ros = ros.fit_resample(x, y)

c. **Undersampling using Tomek Links:** Tomek links are pairs of examples of opposite classes in close vicinity. In this algorithm, we end up removing the majority element from the Tomek link, which provides a better decision boundary for a classifier.



Syntax- from imblearn.under_sampling import TomekLinks
tl = TomekLinks(return_indices=True, ratio='majority')
X_tl, y_tl, id_tl = tl.fit_sample(X, y)

d. **Oversampling using SMOTE:** In SMOTE (Synthetic Minority Oversampling Technique) we synthesize elements for the minority class, in the vicinity of already existing elements.



Syntax: from imblearn.over_sampling import SMOTE
smote = SMOTE(ratio='minority')
X_sm, y_sm = smote.fit_sample(X, y)

There are a variety of other methods in the imblearn package for both undersampling(Cluster Centroids, NearMiss, etc.) and oversampling(ADASYN and bSMOTE)

13. What is the difference between SMOTE and ADASYN sampling techniques?

Ans-

- **SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.**

  ADASYN is a generalized form of the SMOTE algorithm. This algorithm also aims to oversample the minority class by generating synthetic instances for it. But the difference here is it considers the density distribution, ri which decides the no. of synthetic instances generated for samples which difficult to learn. Due to this, it helps in adaptively changing the decision boundaries based on the samples difficult to learn.

- **<u>Working of SMOTE:</u> First it finds the n-nearest neighbours in the minority class for each of the samples in the class. Then it draws a line between the neighbours an generates random points on the lines.**

  <u>Working of ADASYN</u>: ADASYN is a more generic framework, for each of the minority observations it first finds the impurity of the neighbourhood, by taking the ratio of majority observations in the neighbourhood and k.

- **<u>Limitations of SMOTE:</u> If there are observations in the minority class which are outlying and appears in the majority class, it causes a problem for SMOTE, by creating a line bridge with the majority class.**

14. What is the purpose of using GridSearchCV? Is it preferable to use in case of large datasets? Why or why not?

Ans- GridSearchCV is a useful tool to fine tune the parameters of your model (Hyper Parameter Tuning). It runs through all the different parameters that is fed into the parameter grid and produces the best combination of parameters, based on a scoring metric of your choice.

The GridSearchCV class in Sklearn serves a dual purpose in tuning your model by:

1. **Apply a grid search to an array of hyper-parameters, and**
2. **Cross-validate your model using k-fold cross validation**

As Grid Search CV is an exhaustive approach it is not preferable to use in case of large datasets. To go through all combinations of hyperparameters and that too on a very large dataset GridSearchCV can take too much computing resources. And, If you took a whole day to test out parameters and only improved your model accuracy by very little %, perhaps that won't be the best use of your time.

A better use of time with large datasets may be to investigate your features further. Feature engineering and selecting subsets of features can increase (or decrease) the performance of your model tremendously. Even that should be done wisely.

15. List down some of the evaluation metric used to evaluate a regression model. Explain each of them in brief.

Ans- Regression predictive modelling are those problems that involve predicting a numeric value.

The strength of any linear regression model can be assessed using various evaluation metrics. These evaluation metrics usually provide a measure of how well the observed outputs are being generated by the model. These are the metrics that are commonly used for evaluating and reporting the performance of a regression model:

- **Coefficient of Determination or R-Squared (R2)**
- **Mean Squared Error (MSE).**
- **Root Mean Squared Error (RMSE).**
- **Mean Absolute Error (MAE)**

The higher the value of R2, the better is the performance of the model

1. **Coefficient of Determination or R-Squared (R2):** R-Squared is a number that explains the amount of variation that is explained/captured by the developed model. It always ranges between 0 & 1 .

We aim to get a minimum value of below error metrics because this is a loss.

2. **Mean Squared Error:** MSE is an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. We perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

Syntax- from sklearn.metrics import mean_squared_error

```
print("MSE",mean_squared_error(y_test,y_pred))
```

Advantages of MSE

- The graph of MSE is differentiable, so you can easily use it as a loss function.

Disadvantages of MSE

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which can be an advantage in MAE.

3. **Root Mean Squared Error:** The Root Mean Squared Error, or RMSE, is an extension of the mean squared error. Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

We can restate the RMSE in terms of the MSE as: RMSE = sqrt(MSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)}$$

Syntax: print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))

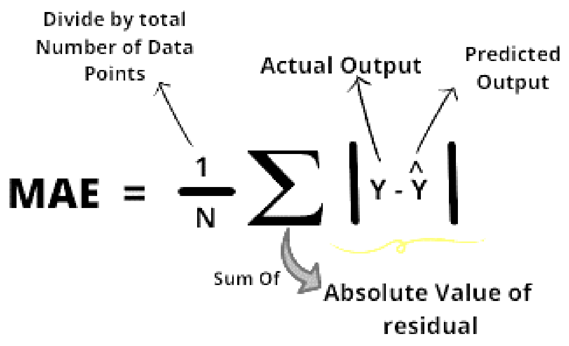
Advantages of RMSE

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.
- Most of the time people use RMSE as an evaluation metric and mostly when working with deep learning techniques the most preferred metric is RMSE.

Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE.

4. **Mean Absolute Error(MAE):** MAE is a very simple metric which calculates the absolute difference between actual and predicted values. We get it by summing all the errors and divide them by a total number of observations.


Divide by total Number of Data Points
Actual Output
Predicted Output
MAE = 1/N ∑ |Y - Ŷ|
Sum Of
Absolute Value of residual

Syntax: from sklearn.metrics import mean_absolute_error
print("MAE",mean_absolute_error(y_test,y_pred))

Advantages of MAE

- The MAE you get is in the same unit as the output variable.
- It is most Robust to outliers.

Disadvantages of MAE

- The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.