WORKSHEET SOLUTIONS- FLIPROBO SET 2

Worksheet-1:

1. The residual sum of squares (RSS) is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model itself. Instead, it estimates the variance in the residuals, or error term.

   Linear regression is a measurement that helps determine the strength of the relationship between a dependent variable and one or more other factors, known as independent or explanatory variables.

Understanding the Residual Sum of Squares

In general terms, the sum of squares is a statistical technique used in regression analysis to determine the dispersion of data points. In a regression analysis, the goal is to determine how well a data series can be fitted to a function that might help to explain how the data series was generated. The sum of squares is used as a mathematical way to find the function that best fits (varies least) from the data.

The RSS measures the amount of error remaining between the regression function and the data set after the model has been run. A smaller RSS figure represents a regression function that is well-fit to the data.

The RSS, also known as the sum of squared residuals, essentially determines how well a regression model explains or represents the data in the model.

How to Calculate the Residual Sum of Squares

$RSS = \sum^{n}_{i=1} (y^{i} - f(x_{i}))^{2}$

y_i = *the* ith *value of the variable to be predicted*

$f(x_i)$ = *predicted value of* y_i
n = *upper limit of summation*

Residual Sum of Squares (RSS) vs. Residual Standard Error (RSE)

The residual standard error (RSE) is another statistical term used to describe the difference in standard deviations of observed values versus predicted values as shown by points in a regression analysis. It is a goodness-of-fit measure that can be used to analyze how well a set of data points fit with the actual model.

RSE is computed by dividing the RSS by the number of observations in the sample less 2, and then taking the square root: RSE = $[\text{RSS}/(n-2)]^{1/2}$

Special Considerations

Financial markets have increasingly become more quantitatively driven; as such, in search of an edge, many investors are using advanced statistical techniques to aid in their decisions. Big data, machine learning, and artificial intelligence applications further necessitate the use of statistical properties to guide contemporary investment strategies. The residual sum of squares—or RSS statistics—is one of many statistical properties enjoying a renaissance.

Statistical models are used by investors and portfolio managers to track an investment's price and use that data to predict future movements. The study—called regression analysis—might involve analyzing the relationship in price movements between a commodity and the stocks of companies engaged in producing the commodity.

2. What is the SST?

The sum of squares total, denoted SST, is the squared differences between the observed *dependent variable* and its mean.

It is a measure of the total variability of the dataset.

Side note: There is another notation for the SST. It is TSS or total sum of squares.

What is the SSR?

The second term is the sum of squares due to regression, or SSR. It is the sum of the differences between the *predicted* value and the mean of the *dependent variable*.

If this value of SSR is equal to the sum of squares total, it means our regression model captures all the observed variability and is perfect. Once again, we have to mention that another common notation is ESS or explained sum of squares.

What is the SSE?

The last term is the sum of squares error, or SSE. The error is the difference between the *observed* value and the *predicted* value.

We usually want to minimize the error. The smaller the error, the better the estimation power of the regression. Finally, I should add that it is also known as RSS or residual sum of squares. Residual as in: remaining or unexplained.

We often use three different sum of squares values to measure how well the regression line actually fits the data:

1. Sum of Squares Total (SST) – The sum of squared differences between individual data points (y_i) and the mean of the response variable ($\boxed{y}$).

- $SST = \Sigma(y_i - \boxed{y})^2$

2. Sum of Squares Regression (SSR) – The sum of squared differences between predicted data points ($\hat{y}_i$) and the mean of the response variable($\boxed{y}$).

- $SSR = \Sigma(\hat{y}_i - \boxed{y})^2$

3. Sum of Squares Error (SSE) – The sum of squared differences between predicted data points ($\hat{y}_i$) and observed data points (y_i).

- $SSE = \Sigma(\hat{y}_i - y_i)^2$

The following relationship exists between these three measures:

SST = SSR + SSE

3. Regularization describes methods for calibrating machine learning models to reduce the adjusted loss function and avoid overfitting or underfitting. We can properly fit our machine learning model on a given test set using regularization, which lowers the errors in the test set.

How Does Regularization Work?

A penalty or complexity term is added to the complex model during regularization. Let’s consider the simple linear regression equation:

y= β0+β1×1+β2×2+β3×3+⋯+βnxn +b

In the above equation, Y represents the value to be predicted

Features for Y are X1, X2, and Xn.

β0,β1,…..βn are the weights or magnitude attached to the features, respectively. Here, stands for the model’s bias, and b stands for the intercept.

Now, in order to create a model that can accurately predict the value of Y, we will add a loss function and optimize a parameter. The loss function for the linear regression is called RSS or residual square sum.

2 Regularization Techniques

Ridge Regularization and Lasso Regularization are the two main categories of regularization techniques.

Ridge Regularization
It is also referred to as Ridge Regression and modifies over- or under-fitted models by adding a penalty equal to the sum of the squares of the coefficient magnitude.

In other words, the mathematical function that represents our machine learning model is minimized, and coefficients are computed. It is squared and added how big the coefficients are. By reducing the number of coefficients, Ridge Regression applies regularization.

Lambda λ is used to represent the penalty term in the cost function. We can control the penalty term by modifying the values of the penalty function. The penalty’s severity affects how much the coefficients are reduced. The parameters are trimmed. In order to avoid multicollinearity, it is used. Additionally, it causes coefficient shrinkage, which lessens the complexity of the model.

Lasso Regression
By introducing a penalty equal to the sum of the absolute values of the coefficients, it modifies the over- or under-fitted models.

Coefficient minimization is also carried out by lasso regression, but the true coefficient values are used rather than the squared coefficient magnitudes. In light of the fact that there are negative coefficients, the coefficient sum can therefore also be 0.

Key Difference Between Ridge Regression And Lasso Regression

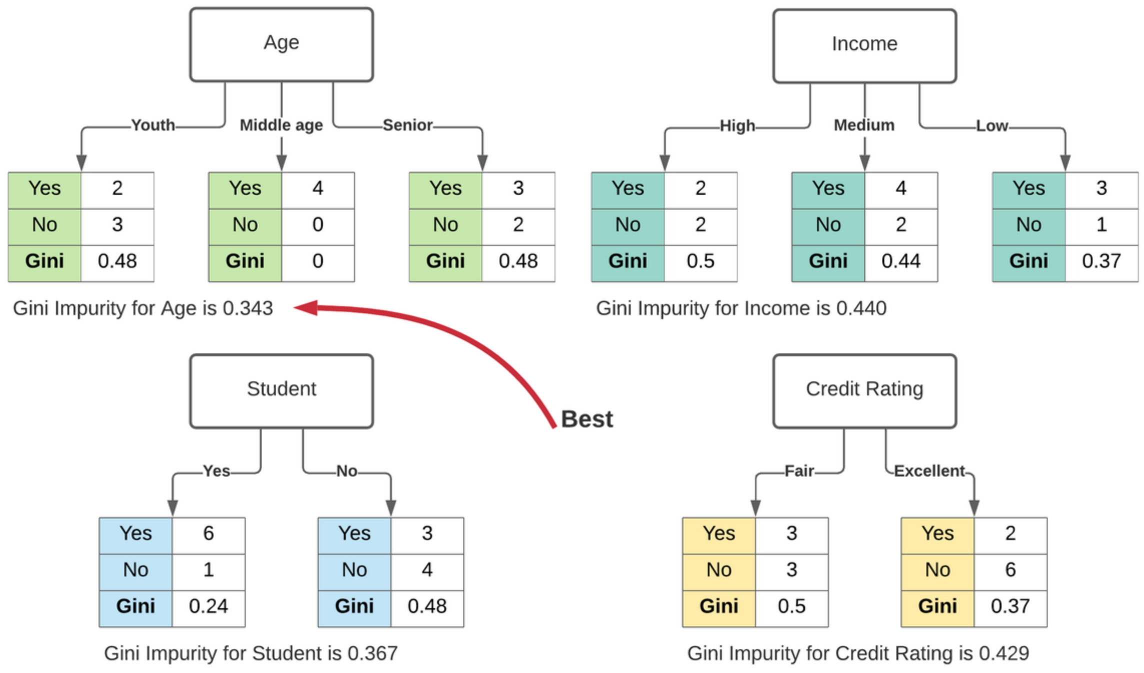
- Ridge regression is mostly used to reduce overfitting in the model, and it includes all the features present in the model. The coefficients are shrunk, which lowers the model's complexity.
- Lasso regression helps to reduce the overfitting in the model as well as feature selection.

What Is The Purpose Of Regularization?

Variance, i.e. variability, is a characteristic of a standard least squares model. this model won't generalize well for a data set different than its training data. Regularization, significantly reduces the variance of the model, without a substantial increase in its bias. Therefore, the regularization techniques described above use the tuning parameter λ to control the effect of bias and variance. As the value of lambda increases, the value of the coefficients decreases, lowering the variance. *Up to a point, this increase in λ is advantageous because it only reduces variance (avoiding overfitting) while maintaining all of the data's significant properties.* But once the value reaches a certain point, the model begins to lose crucial characteristics, leading to bias and underfitting. As a result, care should be taken when choosing the value of &lambda.

You won't require anything more fundamental to start with regularization than this. It is a practical method that can help increase the precision of your regression models. A popular library for implementing these algorithms is Scikit-Learn. It has a wonderful API that can get your model up and running with just a few lines of code in python.

4. Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree. More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.



For example, say you want to build a classifier that determines if someone will default on their credit card. You have some labeled data with features, such as bins for age, income, credit rating, and whether or not each person is a student. To find the best feature for the first split of the tree – the root node – you could calculate how poorly each feature divided the data into the correct class, default ("yes") or didn't default ("no"). This calculation would measure the impurity of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node. This process would continue for each subsequent node using the remaining features.
In the image above, age has minimum gini impurity, so age is selected as the root in the decision tree.

5. Overfitting can be one problem that describes if your model no longer generalizes well. Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it's possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision tree's they can learn a training set to a point of high

granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behavior of this model that makes it prone to learning every point extremely well — to the point of perfect classification — ie: overfitting. I recommend the following steps to avoid overfitting:

- Use a test set that is not exactly like the training set, or different enough that error rates are going to be easy to see. What is different enough? Enough to generalize what is being predicted. The problem should dictate this somewhat because if you are predicting on a baseline that is anomalous, you will need to approximate your validation set close enough. If the problem is more general such as spam classification, having enough data will usually have enough entropy to account for enough variance that should exemplify a disparity is cross validation. Additionally, you can use a one-hold-out dataset for extra assurance. You can be more scientific like using "Minimum Description Length principle" which is something related to the size of the error vs the size of the tree but that's getting a bit in the weeds.
- Ensure you have enough data. It's possible that you don't have enough representitive data (more to my first points in this recommendation). There may not be enough examples to describe a specific case. Perhaps you're classifying houses vs apartments and you don't have enough data on apartments within a given range of values such as square feet and bedrooms, the model may not learn that apartments above 2 bedrooms and 2000 square feet in the city can be either house or apartment but is less likely to be an apartement if there are more houses in the dataset than there are in real life, the decision tree will consider the information gain in this range of variables to describe a split on assumptions it can only conclude with the data it observes. I can't think of a better example…

CHECK FOR CLASS IMBALANCE!

- Reduce the complexity of the decision tree model There's a great quote for this: "Everything should be made as simple as possible, but no simpler." Pruning a tree can be done before or after, but in sklearn, pre-pruning isn't possible, but you can choose to set the minimum amount of data that is required to create a leaf node, which will additionally qualify the conditions on which a split can occur. Additinally, one can set the max-depth to control the complexity, limiting quantity of nodes quite a bit — you could adjust this paramter and check cross-validation each time after you've gridsearched a range that tends to perform well on the classification metric of your choice.
- Use decision trees in an ensemble Since decision trees are greedy, they are also prone to finding locally optimal solutions without considering a broader assumption about data. This is one reason (in my opinion), that makes these ideal for ensembles. Ensemble methods (bagging / random forrests, boosting, etc) that allows for weighting different aspects of decision trees (with bootstrapping, with random variable selection, with weighting of weak learners, with sample weight selection.. these are the main aspects that different ensembles mix and match to generalize better) • Reduce the dimensionality of your data Additionally, choose better features. Reducing the complexity of a model can also be done if you reduce the complexity of your data. The potential splits (ie: complexity), can be reduced before you even put your data to the model.

6. Ensemble learning is a technique in machine learning which takes the help of several base models and combines their output to produce an optimized model. This type of machine learning algorithm helps in improving the overall performance of the model. Here the base model which is most commonly used is the Decision tree classifier. A decision tree basically works on several rules and provides a predictive output, where the rules are the nodes and their decisions will be their children and the leaf nodes will constitute the ultimate decision.

Different types of ensembles, but our major focus will be on the below two types:

- Bagging
- Boosting

These methods help in reducing the variance and bias in a machine learning model. Now let us try to understand what is bias and variance. Bias is an error that occurs due to incorrect assumptions in our algorithm; a high bias indicates our model is too simple/underfit. Variance is the error that is caused due to sensitivity of the model to very small fluctuations in the data set; a high variance indicates our model is highly complex/overfit. An ideal ML model should have a proper balance between bias and variance.

Bootstrap Aggregating/Bagging
Bagging is an ensemble technique that helps in reducing variance in our model and hence avoids overfitting. Bagging is an example of the parallel learning algorithm. Bagging works based on two principles.

- Bootstrapping: From the original data set, different sample populations are considered with replacement.
- Aggregating: Averaging out the results of all the classifiers and providing single output, for this, it uses majority voting in the case of classification and averaging in the case of the regression problem. One of the famous

machine learning algorithms which use the concept of bagging is a random forest.

Random Forest

In random forest from the random sample withdrawn from the population with replacement and a subset of features is selected from the set of all the features a decision tree is built. From these subsets of features whichever feature gives the best split is selected as the root for the decision tree. The features subset must be chosen randomly at any cost otherwise we will end up producing only correlated tress and the variance of the model will not be improved.

Now we have built our model with the samples taken from the population, the question is how do we validate the model? Since we are considering the samples with replacement hence all the samples will not be considered and some of it will not be included in any bag these are called out of bag samples. We can validate our model with this OOB (out of bag) samples. The important parameters to be considered in a random forest is the number of samples and the number of trees. Let us consider 'm' as the subset of features and 'p' is the full set of features, now as a thumb rule, it's always ideal to choose

Boosting

Boosting is a sequential learning algorithm that helps in reducing bias in our model and variance in some cases of supervised learning. It also helps in converting weak learners into strong learners. Boosting works on the principle of placing the weak learners sequentially and it assigns a weight to each data point after every round; more weight is assigned to the misclassified data point in the previous round. This sequential weighted method of training our data set is the key difference to that of bagging.

7. Difference between Bagging and Boosting

S.NO	Bagging	Boosting
1.	Bagging is a learning approach that aids in enhancing the performance, execution, and precision of machine learning algorithms.	Boosting is an approach that iteratively modifies the weight of observation based on the last classification.
2.	It is the easiest method of merging predictions that belong to the same type.	It is a method of merging predictions that belong to different types.
3.	Here, every model has equal weight.	Here, the weight of the models depends on their performance.
4.	In bagging, each model is assembled independently.	In boosting, the new models are impacted by the implementation of earlier built models.
5.	It helps in solving the over-fitting issue.	It helps in reducing the bias.
6.	In the case of bagging, if the classifier is unstable, then we apply bagging.	In the case of boosting, If the classifier is stable, then we apply boosting.

8. The out-of-bag error is the average error for each predicted outcome calculated using predictions from the trees that do not contain that data point in their respective bootstrap sample. This way, the Random Forest model is constantly being validated while being trained. Let us consider the *j*th decision tree DTj that has been fitted on a subset of the sample data. For every training

observation or sample zi=(xi,yi) not in the sample subset of DTj where xi is the set of features and yi is the target, we use to predict the outcome oi for xi. The error can easily be computed as |oi−yi|.
The out-of-bag error is thus the average value of this error across all decision trees.

9. For evaluating a model's performance and hyperparameter tuning *k-fold cross-validation* is one of the most popular strategies widely used by data scientists. It is a *data partitioning strategy* so that you can effectively use your dataset to build *a more generalized model*. The main intention of doing any kind of machine learning is to develop a more generalized model which can perform well on *unseen data*. One can build a perfect model on the training data with 100% accuracy or 0 error, but it may fail to generalize for unseen data. So, it is not a good model. It overfits the training data. Machine Learning is all about *generalization* meaning that model's performance can only be measured with data points that have never been used during the training process. That is why we often split our data into a training set and a test set.

Data splitting process can be done more effectively with k-fold cross-validation. Here, we discuss two scenarios which involve k-fold cross-validation. Both involve splitting the dataset, but with different approaches.

- Using k-fold cross-validation for evaluating a model's performance
- Using k-fold cross-validation for hyperparameter tuning

10. Hyperparameter tuning in machine learning is a technique where we tune or change the default parameters of the existing model or algorithm to achieve higher accuracies and better performance. Sometimes when we use the default parameters of the algorithms, it does not suit the existing data as the data can vary according to the problem statement. In that case, the hyperparameter tuning becomes an essential part of the model building to enhance the model's performance.

This article will discuss the algorithm's hyperparameter tuning, advantages, and other related things. This will help one understand the concept of hyperparameter tuning and its need and help one perform it for any kind of data and model.

What is the Need for Hyperparameter Tuning

In machine learning, the behavior and patterns of every data cannot be the same, and the problem statement that we are working with also varies. Almost all machine learning algorithms have their default parameters, which will be applied if no specific parameters are selected.

In every case or project, the default parameters of the machine learning algorithm can not be the best suitable solution for us; the model may perform well with the default parameters but in some cases, the performance and reliability can still be increased by tuning those parameters.

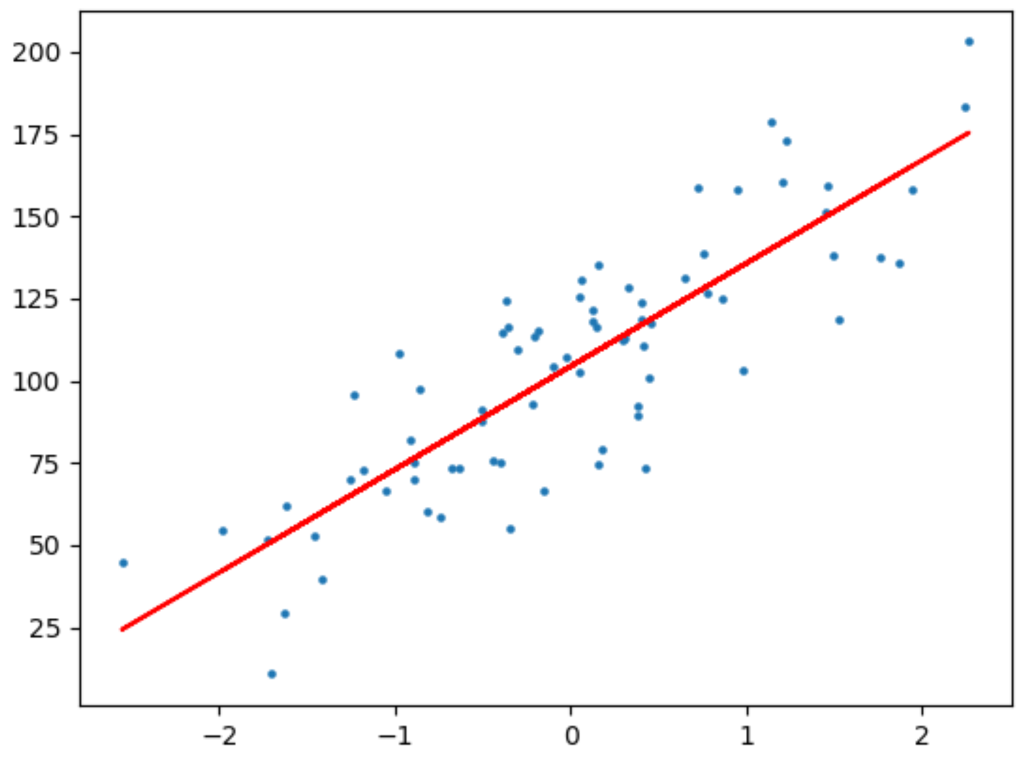
We can tune the parameters of the algorithm as per our need, the type of data, the behavior of the data, data patterns, and the target we want to achieve from the model.

11. *Learning rate is one of The most important thing to consider in whole of machine learning. But choosing the correct learning rate is pretty much impossible all the time.*

But…, having the knowledge of what it can do is useful in some cases where we can at least guess what could be the appropriate learning rate.

We will see how even the slightest change in learning_rate can improve the speed drastically or could break your model. I mean it can overshoot the minima and it might never return .

```
X, y = make_regression(n_samples=75, n_features=1, random_state=0, noise=20,
bias=100)
```



- We will draw the regression line(in red) with learned Linear Regression and we put that as a benchmark.

- We choose Linear Regression over any other because it is easy to understand (and easy to code too).

- we will focus mainly on how learning rate effects on optimizing least squared error of Linear Regression.

Note: Here we won't be explaining linear regression in much detail.

Since we have 1-D data, we can easily fit a line to it. (We have to predict y given X).

- This line is what we call '*weight vector*' or '*hypothesis*'.

- By using this line (the plane in higher dimensional data), we will fit the line with minimum error possible. We have only one x and output for that x. We will try to fit the line/plane that is best for all the data points in this dataset.

- After we find the best line for this data, we will predict y given any new x.

Let's say our equation of our line is :

$$w = w_0 + w_1 x_1$$

$$w_0 : y - intercept$$

$$w_1 : slope$$

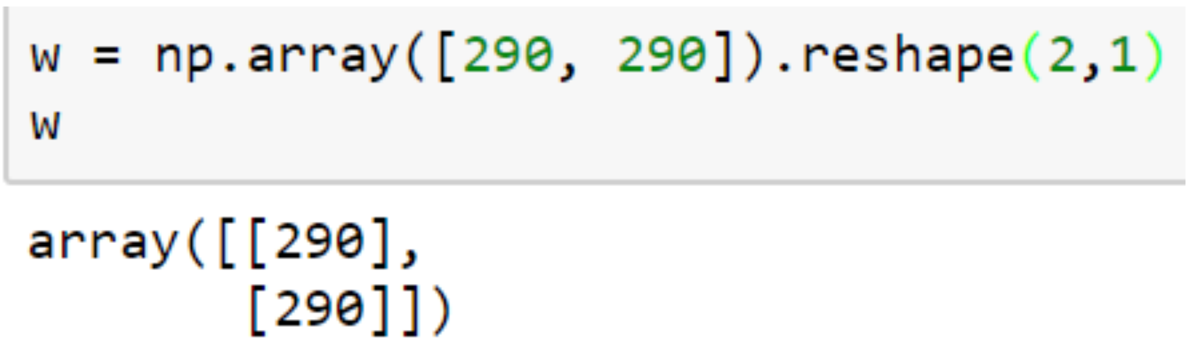
- After we find the best line for this data, given any new x, we will just substitute our data (x_1 in this case), in the equation which is optimized with better w_0 and w_1 values, to predict y.

To make our calculation simpler (calculating the Gradient Descent with vector multiplication rather than traditional for loops..), we will rewrite the equation as :

$w = w_0 x_0 + w_1 x_1$, with x_0 as always 1.

Initial Weights :

The initial weights should be random. But we chose some specific numbers just to make sure that is not anywhere near the Optimal Weights. Reshaping is done to make it easier while computing cost.



```
w = np.array([290, 290]).reshape(2,1)
w
```

```
array([[290],
       [290]])
```

Initial weights (W_0, W_1)

12. Logistic regression is known and used as a linear classifier. It is used to come up with a hyper*plane* in feature space to separate observations that belong to a class from all the other observations that do *not* belong to that class. The decision boundary is thus *linear*. Robust and efficient implementations are readily available (e.g. scikit-learn) to use logistic regression as a linear classifier. While logistic regression makes core assumptions about the observations such as IID (each observation is independent of the others and they all have an identical probability distribution), the use of a linear decision boundary is *not* one of them. The linear decision boundary is used for reasons of simplicity following the Zen mantra — when in doubt simplify. In those cases where we suspect the decision boundary to be nonlinear, it may make sense to formulate logistic regression with a nonlinear model and evaluate how much better we can do.
13. The Comparison:

Loss Function:
The technique of Boosting uses various loss functions. In case of Adaptive Boosting or AdaBoost, it minimises the exponential loss function that can make the algorithm sensitive to the outliers. With Gradient Boosting, any differentiable loss function can be utilised. Gradient Boosting algorithm is more robust to outliers than AdaBoost.

Flexibility
AdaBoost is the first designed boosting algorithm with a particular loss function. On the other hand, Gradient Boosting is a generic algorithm that

assists in searching the approximate solutions to the additive modelling problem. This makes Gradient Boosting more flexible than AdaBoost.

Benefits
AdaBoost minimises loss function related to any classification error and is best used with weak learners. The method was mainly designed for binary classification problems and can be utilised to boost the performance of decision trees. Gradient Boosting is used to solve the differentiable loss function problem. The technique can be used for both classification and regression problems.

Shortcomings
In the case of Gradient Boosting, the shortcomings of the existing weak learners can be identified by gradients and with AdaBoost, it can be identified by high-weight data points.

Wrapping Up
Though there are several differences between the two boosting methods, both the algorithms follow the same path and share similar historic roots. Both the algorithms work for boosting the performance of a simple base-learner by iteratively shifting the focus towards problematic observations that are challenging to predict.

In the case of AdaBoost, the shifting is done by up-weighting observations that were misclassified before, while Gradient Boosting identifies the difficult observations by large residuals computed in the previous iterations.

14. The bias-variance tradeoff is a fundamental concept in machine learning and statistics that relates to the ability of a model to accurately capture the underlying patterns in a dataset. In essence, the bias-variance tradeoff refers to the balance between the complexity of a model and its ability to generalize to new, unseen data. The bias-variance tradeoff arises because it is difficult to simultaneously minimize both bias and variance. As the complexity of a model increases, its bias decreases, but its variance increases. Conversely, as the complexity of a model decreases, its bias increases, but its variance decreases. The optimal level of

complexity for a model depends on the specific problem and the available data.

15. Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface is able to transform to a linear equation in a higher number of dimension spaces. Basically, It returns the inner product between two points in a standard feature dimension.

    Polynomial Kernel: It represents the similarity of vectors in the training set of data in a feature space over polynomials of the original variables used in the kernel. K (x, y) = tanh (\gamma.{x^T y}+{r})^d, \gamma>0

    Linear Kernel: used when data is linearly separable.
    Gaussian Kernel: It is used to perform transformation when there is no prior knowledge about data.
    K (x, y) = e ^ - (\frac{||x - y||^2} {2 \sigma^2})

    Gaussian Kernel Radial Basis Function (RBF): Same as above kernel function, adding radial basis method to improve the transformation.
    K (x, y) = e ^ - (\gamma{||x - y||^2}) K (x, x1) + K (x, x2) (Simplified - Formula) K (x, x1) + K (x, x2) > 0 (Green) K (x, x1) + K (x, x2) = 0 (Red)

Worksheet-2:

1. SELECT * FROM movie
2. SELECT runtime FROM movie WHERE ((movie.movie_ID)=[MovieGenre].[Movie]) GROUP BY tblMovie.runtime;
3. SELECT revenue FROM movie WHERE revenue ("highest revenue")
4. SELECT title FROM movie WHERE revenue ("highest revenue")
5. SELECT title FROM movie WHERE cast_details(person, gender, character name, cast order.);
6. SELECT title, rating FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie_id ORDER BY rating DESC;
7. SELECT movies.*, genres.name FROM movies INNER JOIN genres_in_movies ON genres_in_movies.movie_id = movies.id INNER JOIN genres ON genres_in_movies.genre_id = genres.id GROUP BY movies.title;
8. SELECT dir_fname,dir_lname, gen_title,count(gen_title)
   ```
   FROM director
   NATURAL JOIN movie_direction
   NATURAL JOIN movie_genres
   NATURAL JOIN genres
   GROUP BY dir_fname, dir_lname,gen_title
   ORDER BY dir_fname,dir_lname;
   ```
9. SELECT mov_title, mov_year, gen_title, dir_fname, dir_lname
   ```
   FROM movie
   NATURAL JOIN movie_genres
   NATURAL JOIN genres
   NATURAL JOIN movie_direction
   NATURAL JOIN director;
   ```
10. SELECT movie.mov_title, mov_year, mov_dt_rel, mov_time,dir_fname, dir_lname
    ```
    FROM movie
    JOIN  movie_direction
      ON movie.mov_id = movie_direction.mov_id
    JOIN director
      ON movie_direction.dir_id=director.dir_id
    WHERE mov_dt_rel <'01/01/1989'
    ORDER BY mov_dt_rel desc;
    ```
11. SELECT title as MovieName, revenue from (select * from movie order by revenue desc limit 3);
12. SELECT title as MovieName FROM movie

    WHERE movie_id = (SELECT movie_id FROM movie WHERE movie_status = ' rumoured ');
13. SELECT title as MovieName ,MAX(revenue) FROM movie WHERE movie_id = (SELECT movie_id FROM movie LEFT JOIN production_country ON movie.movie_id= production_country.movie_id
    LEFT JOIN country ON
    country. country _id= production_country. country _id
    WHERE country_name = ' United States of America ');
14. SELECT movie_company.movie_id,production_company.company_name FROM movie_company RIGHTJOIN production_company ON movie_company.company_id= production_company.company_id;
15. select title from (select * from movie order by budget desc limit 20);

Worksheet-3:

1. D) Expected
2. C) Frequencies
3. C) 6
4. A) Normal distribution
5. C) F- Distribution
6. B) Hypothesis
7. A) Null Hypothesis
8. A) Two tailed
9. B) Research Hypothesis
10. A) np