STATISTICS WORKSHEET-5

1. Using a goodness of fit, we can assess whether a set of obtained frequencies differ from a set of frequencies.
a) Mean
b) Actual
c) Predicted
d) Expected

2. Chisquare is used to analyse
a) Score
b) Rank
c) Frequencies
d) All of these

3. What is the mean of a Chi Square distribution with 6 degrees of freedom?
a) 4
b) 12
c) 6
d) 8

4. Which of these distributions is used for a goodness of fit testing?
a) Normal distribution
b) Chisqared distribution
c) Gamma distribution
d) Poission distribution

5. Which of the following distributions is Continuous
a) Binomial Distribution
b) Hypergeometric Distribution
c) F Distribution
d) Poisson Distribution

6. A statement made about a population for testing purpose is called?
a) Statistic
b) Hypothesis
c) Level of Significance
d) TestStatistic

7. If the assumed hypothesis is tested for rejection considering it to be true is called?
a) Null Hypothesis
b) Statistical Hypothesis
c) Simple Hypothesis
d) Composite Hypothesis

8. If the Critical region is evenly distributed then the test is referred as?

a) Two tailed

b) One tailed

c) Three tailed

d) Zero tailed

9. Alternative Hypothesis is also called as?

a) Composite hypothesis

b) Research Hypothesis

c) Simple Hypothesis

d) Null Hypothesis

10. In a Binomial Distribution, if 'n' is the number of trials and 'p' is the probability of success, then the mean value is given by

a) np

b) n

MACHINE LEARNING

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

Ans:

1. Total sum of squares

The total sum of squares is a variation of the values of a dependent variable from the sample mean of the dependent variable. Essentially, the total sum of squares quantifies the total variation in a sample. It can be determined using the following formula:

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where:

- y_i – the value in a sample
- $\bar{y}$ – the mean value of a sample

2. Regression sum of squares (also known as the sum of squares due to regression or explained sum of squares)

The regression sum of squares describes how well a regression model represents the modeled data. A higher regression sum of squares indicates that the model does not fit the data well.

The formula for calculating the regression sum of squares is:

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

Where:

- $\hat{y}_i$ – the value estimated by the regression line
- $\bar{y}$ – the mean value of a sample

3. Residual sum of squares (also known as the sum of squared errors of prediction)

The residual sum of squares essentially measures the variation of modeling errors. In other words, it depicts how the variation in the dependent variable in

a regression model cannot be explained by the model. Generally, a lower residual sum of squares indicates that the regression model can better explain the data, while a higher residual sum of squares indicates that the model poorly explains the data.

The residual sum of squares can be found using the formula below:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Where:

- y_i – the observed value
- $\hat{y}_i$ – the value estimated by the regression line

The relationship between the three types of sum of squares can be summarized by the following equation:

$$TSS = SSR + SSE$$

3. What is the need of regularization in machine learning?

Ans : Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding extra information to it.

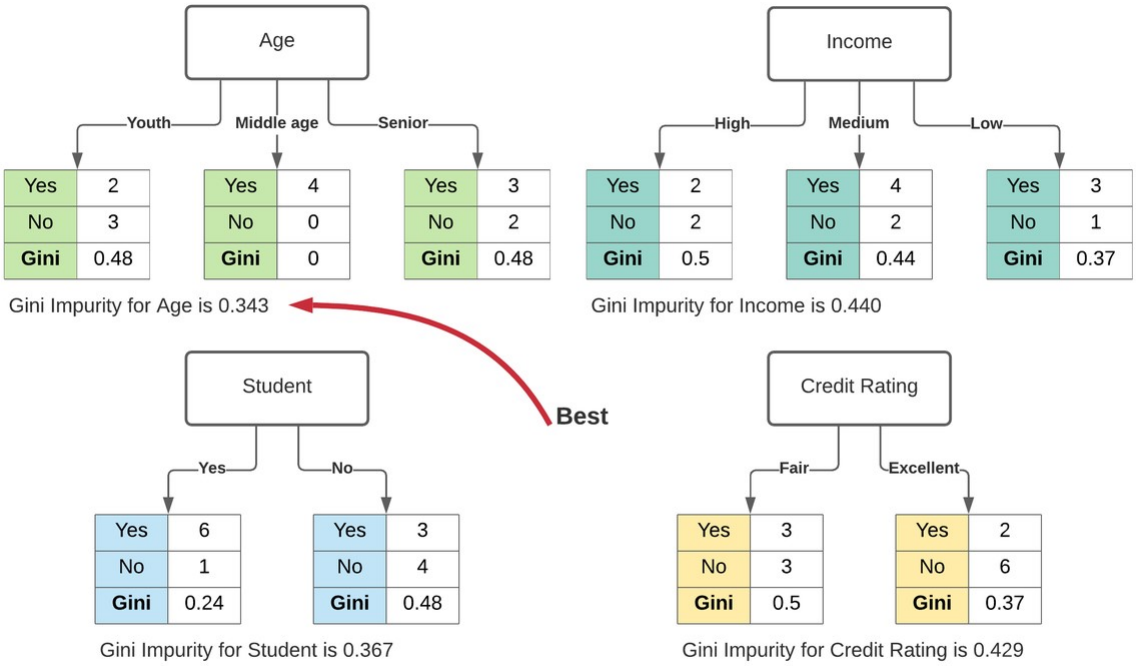
Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "*In regularization technique, we reduce the magnitude of the features by keeping the same number of features.*"

4. What is Gini–impurity index?

Ans: 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. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans :

Over-fitting is the phenomenon in which the learning system tightly fits the given training data so much that it would be inaccurate in predicting the outcomes of the untrained data.

In decision trees, over-fitting occurs when the tree is designed so as to perfectly fit all samples in the training data set. Thus it ends up with branches with strict rules of sparse data. Thus this effects the accuracy when predicting samples that are not part of the training set.

One of the methods used to address over-fitting in decision tree is called **pruning** which is done after the initial training is complete. In pruning, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and

validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.

Details of decision tree over-fitting and pruning is given in the book *Machine Learning* by *Tom Mitchell.* It should be simple for beginners to understand.

https://gist.github.com/dyerrington/b136a24e4137415b307fde68aa8cb53b

6. What is an ensemble technique in machine learning?

Ans:

ensemble algorithm, as well as critical ensemble techniques, such as boosting and bagging. But before digging deep into the what, why, and how of ensemble, let's first take a look at some real-world examples that will simplify the concepts that are at the core of ensemble learning.

Example 1: If you are planning to buy an air-conditioner, would you enter a showroom and buy the air-conditioner that the salesperson shows you? The answer is probably no. In this day and age, you are likely to ask your friends, family, and colleagues for an opinion, do research on various portals about different models, and visit a few review sites before making a purchase decision. In a nutshell, you would not come to a conclusion directly. Instead, you would try to make a more informed decision after considering diverse opinions and reviews. In the case of ensemble learning, the same principle applies. Now let's see what ensemble means.

The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions. These methods follow the same principle as the example of buying an air-conditioner cited above.

In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms.

Example 2: Assume that you are developing an app for the travel industry. It is obvious that before making the app public, you will want to get crucial feedback on bugs and potential loopholes that are affecting the user experience. What are your available options for obtaining critical feedback? 1) Soliciting opinions from your parents, spouse, or close friends. 2) Asking your co-workers who travel regularly and then evaluating their response. 3) Rolling out your travel and tourism app in beta to gather feedback from non-biased audiences and the travel community.

Think for a moment about what you are doing. You are taking into account different views and ideas from a wide range of people to fix issues that are limiting the user experience. The ensemble neural network and ensemble algorithm do precisely the same thing.

Example 3: Imagine a group of blindfolded people playing the touch-and-tell game, where they are asked to touch and explore a mini donut factory that no one of them has ever seen before. Since they are blindfolded, their version of

what a mini donut factory looks like will vary, depending on the parts of the appliance they touch. Now, suppose they are personally asked to describe what they touched. In that case, their individual experiences will give a precise description of specific parts of the mini donut factory. Still, collectively, their combined experiences will provide a highly detailed account of the entire equipment.

Similarly, ensemble methods in machine learning employ a set of models and take advantage of the blended output, which, compared to a solitary model, will most certainly be a superior option when it comes to prediction accuracy.

Ensemble Techniques

Here is a list of ensemble learning techniques, starting with basic ensemble methods and then moving on to more advanced approaches.

Simple Ensemble Methods

Mode: In statistical terminology, "mode" is the number or value that most often appears in a dataset of numbers or values. In this ensemble technique, machine learning professionals use a number of models for making predictions about each data point. The predictions made by different models are taken as separate votes. Subsequently, the prediction made by most models is treated as the ultimate prediction.

The Mean/Average: In the mean/average ensemble technique, data analysts take the average predictions made by all models into account when making the ultimate prediction.

Let's take, for instance, one hundred people rated the beta release of your travel and tourism app on a scale of 1 to 5, where 15 people gave a rating of 1, 28 people gave a rating of 2, 37 people gave a rating of 3, 12 people gave a rating of 4, and 8 people gave a rating of 5.

The average in this case is - (1 * 15) + (2 * 28) + (3 * 37) + (4 * 12) + (5 * 8) / 100 = 2.7

The Weighted Average: In the weighted average ensemble method, data scientists assign different weights to all the models in order to make a prediction, where the assigned weight defines the relevance of each model. As an example, let's assume that out of 100 people who gave feedback for your travel app, 70 are professional app developers, while the other 30 have no experience in app development. In this scenario, the weighted average ensemble technique will give more weight to the feedback of app developers compared to others.

Advanced Ensemble Methods

Bagging (Bootstrap Aggregating): The primary goal of "bagging" or "bootstrap aggregating" ensemble method is to minimize variance errors in decision trees. The objective here is to randomly create samples of training datasets with replacement (subsets of the training data). The subsets are then used for training decision trees or models. Consequently, there is a combination of multiple models, which reduces variance, as the average prediction generated

from different models is much more reliable and robust than a single model or a decision tree.

Boosting: An iterative ensemble technique, "boosting," adjusts an observation's weight based on its last classification. In case observation is incorrectly classified, "boosting" increases the observation's weight, and vice versa. Boosting algorithms reduce bias errors and produce superior predictive models.

In the boosting ensemble method, data scientists train the first boosting algorithm on an entire dataset and then build subsequent algorithms by fitting residuals from the first boosting algorithm, thereby giving more weight to observations that the previous model predicted inaccurately.

Interested in Mastering Ensemble Algorithms for a Rewarding Career in Machine Learning?

The easiest way to secure a high-paying machine learning job is to get yourself certified from a globally renowned educational institution, such as Simplilearn. The Post Graduate Program in AI and Machine Learning, introduced by the world's #1 online bootcamp and certification course provider, Simplilearn, in collaboration with internationally famed Purdue University and IBM, will provide you with an in-depth understanding of the core concepts of ensemble methods in machine learning.

Ref : https://www.simplilearn.com/ensemble-learning-article

7. What is the difference between Bagging and Boosting techniques?

Ans :

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.

S.NO	Bagging	Boosting
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

Ref : https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/

8. What is out-of-bag error in random forests?

Ans: The *out-of-bag* (OOB) error is the average error for each calculated using predictions from the trees that do not contain in their respective bootstrap sample. This allows the RandomForestClassifier to be fit and validated whilst being trained [1].

What is the Out of Bag score in Random Forests?

Out of bag (OOB) score is a way of validating the Random forest model. Below is a simple intuition of how is it calculated followed by a description of how it is different from validation score and where it is advantageous.

For the description of OOB score calculation, let's assume there are five DTs in the random forest ensemble labeled from 1 to 5. For simplicity, suppose we have a simple original training data set as below.

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Let the first bootstrap sample is made of the first three rows of this data set as shown in the green box below. This bootstrap sample will be used as the training data for the DT "1".

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Bootstrap sample

Then the last row that is "left out" in the original data (see the red box in the image below) is known as Out of Bag sample. This row will not be used as the training data for DT 1. Please note that in reality there will be several such rows which are left out as Out of Bag, here for simplicity only one is shown.

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Out of Bag sample

After the DTs models have been trained, this leftover row or the OOB sample will be given as unseen data to the DT 1. The DT 1 will predict the outcome of this row. Let DT 1 predicts this row correctly as "YES". Similarly, this row will be passed through all the DTs that did not contain this row in their bootstrap training data. Let's assume that apart from DT 1, DT 3 and DT 5 also did not have this row in their bootstrap training data. The predictions of this row by DT 1, 3, 5 are summarized in the table below.

Decision Tree	Prediction
1	YES
3	NO
5	YES
Majority vote : YES	

We see that by a majority vote of 2 “YES” vs 1 “NO” the prediction of this row is “YES”. It is noted that the final prediction of this row by majority vote is a **correct prediction** since originally in the “Play Tennis” column of this row is also a “YES”.

Similarly, each of the OOB sample rows is passed through every DT that did not contain the OOB sample row in its bootstrap training data and a majority prediction is noted for each row.

And lastly, the OOB score is computed as **the number of correctly predicted rows from the out of bag sample.**

Ref : https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710

9. What is K-fold cross-validation?

Ans K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into. For example, if you see that the k-value is 5, we can call this a 5-fold cross-validation. Each fold is used as a testing set at one point in the process.

K-fold Cross-Validation Process:

1. Choose your k-value
2. Split the dataset into the number of k folds.
3. Start off with using your k-1 fold as the test dataset and the remaining folds as the training dataset
4. Train the model on the training dataset and validate it on the test dataset
5. Save the validation score
6. Repeat steps 3 – 5, but changing the value of your k test dataset. So we chose k-1 as our test dataset for the first round, we then move onto k-2 as the test dataset for the next round.
7. By the end of it you would have validated the model on every fold that you have.

8. Average the results that were produced in step 5 to summarize the skill of the model.

Why Should I Use K-fold Cross-Validation?

I am going to break down the reasons below as to why you should use K-fold Cross Validation

Make use of your data

We have a lot of readily available data that can explain a lot of things and help us identify hidden patterns. However, if we only have a small dataset, splitting it into a training and test dataset at a 80:20 ratio respectively doesn't seem to do much for us.

However, when using K-fold cross validation, all parts of the data will be able to be used as part of the testing data. This way, all of our data from our small dataset can be used for both training and testing, allowing us to better evaluate the performance of our model.

Ref: https://www.kdnuggets.com/2022/07/kfold-cross-validation.html#:~:text=K%2Dfold%20Cross%2DValidation%20is,5%2Dfold%20cross%2Dvalidation.

10. What is hyper parameter tuning in machine learning and why it is done?

Ans :

Hyperparameter tuning is an essential part of controlling the behavior of a machine learning model. If we don't correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don't minimize the loss function. This means our model makes more errors. In practice, key indicators like the accuracy or the confusion matrix will be worse.

What are hyperparameters?

In machine learning, we need to differentiate between parameters and hyperparameters. A learning algorithm learns or estimates model parameters for the given data set, then continues updating these values as it continues to learn. After learning is complete, these parameters become part of the model. For example, each weight and bias in a neural network is a parameter.

Hyperparameters, on the other hand, are specific to the algorithm itself, so we can't calculate their values from the data. We use hyperparameters to calculate the model parameters. Different hyperparameter values produce different model parameter values for a given data set.

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors. Note that the learning algorithm optimizes the loss based on the input data and tries to find an optimal solution within the given setting. However, hyperparameters describe this setting exactly.

Hyperparameter types

Some important hyperparameters that require tuning in neural networks are:

- **Number of hidden layers**: It's a trade-off between keeping our neural network as simple as possible (fast and generalized) and classifying our input data correctly. We can start with values of four to six and check our data's prediction accuracy when we increase or decrease this hyperparameter.

- **Number of nodes/neurons per layer**: More isn't always better when determining how many neurons to use per layer. Increasing neuron count can help, up to a point. But layers that are too wide may memorize the training dataset, causing the network to be less accurate on new data.

- **Learning rate**: Model parameters are adjusted iteratively — and the learning rate controls the size of the adjustment at each step. The lower the learning rate, the lower the changes to parameter estimates are. This means that it takes a longer time (and more data) to fit the model — but it also means that it is more likely that we actually find the minimum loss.

- **Momentum**: Momentum helps us avoid falling into local minima by resisting rapid changes to parameter values. It encourages parameters to keep changing in the direction they were *already* changing, which helps prevent zig-zagging on every iteration. Aim to start with low momentum values and adjust upward as needed.

We consider these essential hyperparameters for tuning SVMs:

- **C**: A trade-off between a smooth decision boundary (more generic) and a neat decision boundary (more accurate for the training data). A low value may cause the model to incorrectly classify some training data, while a high value may cause the model to incur overfitting. Overfitting creates an analysis too specific for the current data set and possibly unfit for future data and unreliable for future observations.

- **Gamma**: The inverse of the influence radius of data samples we selected as support vectors. High values indicate the small radius of influence and small decision boundaries that do not consider relatively close data samples. These high values cause overfitting. Low values indicate the significant effect of distant data samples, so the model can't capture the correct decision boundaries from the data set.

Methods for tuning hyperparameters

Now that we understand what hyperparameters are and the importance of tuning them, we need to know how to choose their optimal values. We can find these optimal hyperparameter values using manual or automated methods.

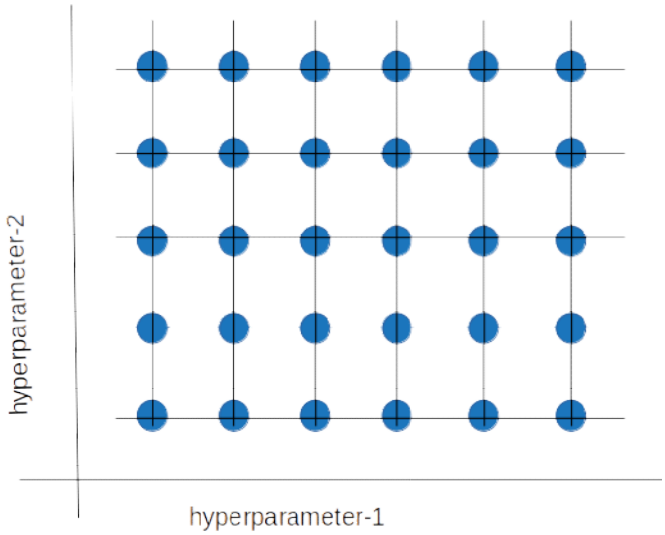
When tuning hyperparameters manually, we typically start using the default recommended values or rules of thumb, then search through a range of values using trial-and-error. But manual tuning is a tedious and time-consuming

approach. It isn't practical when there are many hyperparameters with a wide range.

Automated hyperparameter tuning methods use an algorithm to search for the optimal values. Some of today's most popular automated methods are grid search, random search, and Bayesian optimization. Let's explore these methods in detail.

Grid search

Grid search is a sort of "brute force" hyperparameter tuning method. We create a grid of possible discrete hyperparameter values then fit the model with every possible combination. We record the model performance for each set then select the combination that has produced the best performance.



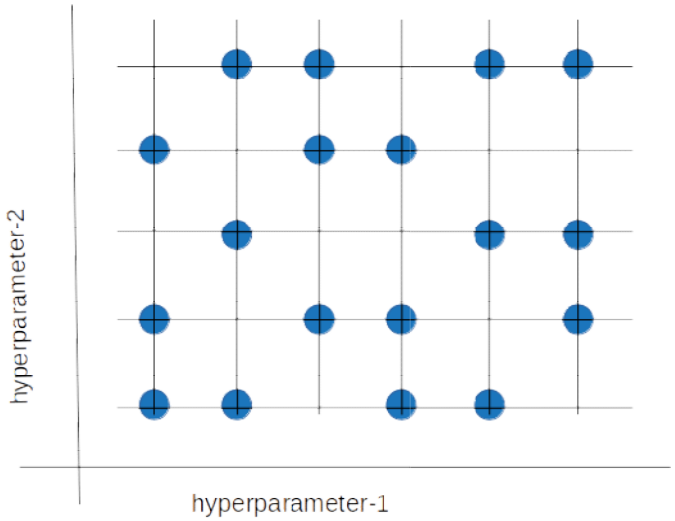
Grid search is a hyperparameter tuning method in which we create a grid of possible discrete hyperparameter values, then fit the model with every possible combination.

Grid search is an exhaustive algorithm that can find the best combination of hyperparameters. However, the drawback is that it's slow. Fitting the model with every possible combination usually requires a high computation capacity and significant time, which may not be available.

Random search

The random search method (as its name implies) chooses values randomly rather than using a predefined set of values like the grid search method.

Random search tries a random combination of hyperparameters in each iteration and records the model performance. After several iterations, it returns the mix that produced the best result.



Random search tries a random combination of hyperparameters in each iteration and records the model performance. After several iterations, it returns the mix that produced the best result.

Random search is appropriate when we have several hyperparameters with relatively large search domains. We can make discrete ranges (for instance, [5-100] in steps of 5) and still get a reasonably good set of combinations.

The benefit is that random search typically requires less time than grid search to return a comparable result. It also ensures we don't end up with a model that's biased toward value sets arbitrarily chosen by users. Its drawback is that the result may not be the best possible hyperparameter combination.

Bayesian optimization

Grid search and random search are relatively inefficient because they often evaluate many unsuitable hyperparameter combinations. They don't take into account the previous iterations' results when choosing the next hyperparameters.

The Bayesian optimization method takes a different approach. This method treats the search for the optimal hyperparameters as an optimization problem. When choosing the next hyperparameter combination, this method considers the previous evaluation results. It then applies a probabilistic function to select the combination that will probably yield the best results. This method discovers a fairly good hyperparameter combination in relatively few iterations.

Data scientists choose a probabilistic model when the objective function is unknown. That is, there is no analytical expression to maximize or minimize. The data scientists apply the learning algorithm to a data set, use the algorithm's results to define the objective function, and take the various hyperparameter combinations as the input domain.

The probabilistic model is based on past evaluation results. It estimates the probability of a hyperparameter combination's objective function result:

P(result | hyperparameters)

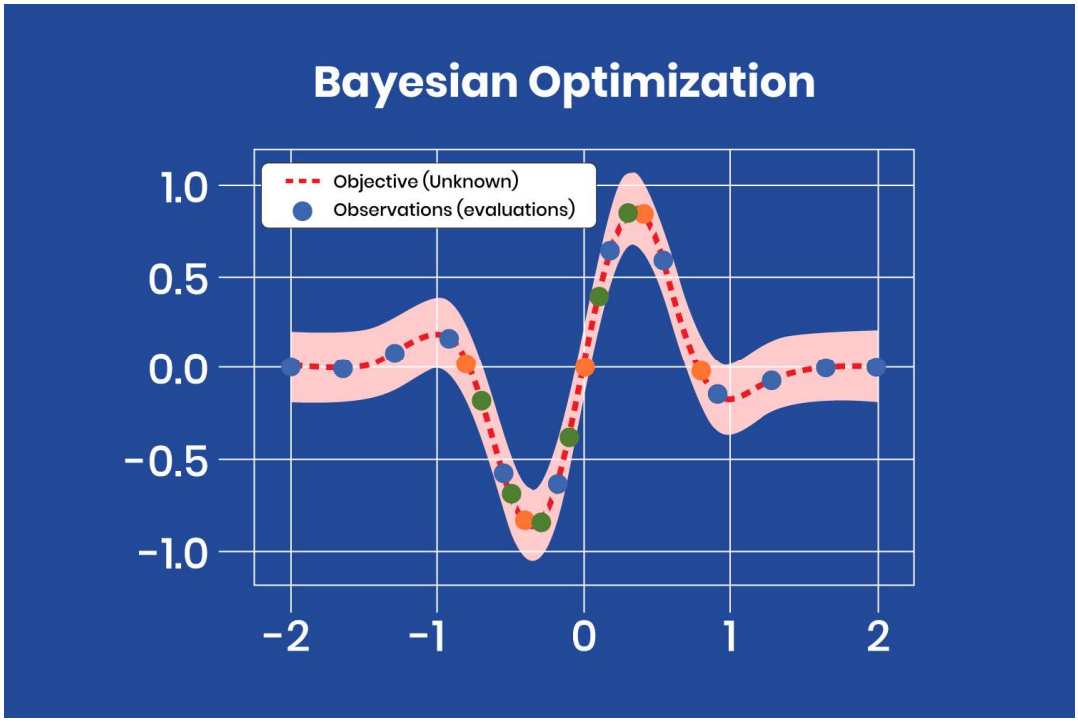
This probabilistic model is a "surrogate" of the objective function. The objective function can be, for instance, the root-mean-square error (RMSE). We calculate

the objective function using the training data with the hyperparameter combination. We try to optimize it (maximize or minimize, depending on the objective function selected).

Applying the probabilistic model to the hyperparameters is computationally inexpensive compared to the objective function, so this method typically updates and improves the surrogate probability model every time the objective function runs. Better hyperparameter predictions decrease the number of objective function evaluations we need to achieve a good result.

Gaussian processes, random forest regression, and tree-structured Parzen estimators (TPE) are surrogate model examples.

The figure below shows how a surrogate function finds the minimum of the "objective" function, where the "objective" function is unknown.



The Bayesian optimization method treats the search for the optimal hyperparameters as an optimization problem.

After a few iterations, with the evaluations (observations) obtained from the "objective" function, the surrogate model finds the minimum at the coordinates x=-0.35 and y=-0.8. Note that there are several evaluations close to that point, corresponding to the latest iterations, as the model has almost found the minimum and produces similar values (different color points correspond to different iterations).

The Bayesian optimization model is complex to implement. Fortunately, we can use off-the-shelf libraries like Ray Tune to simplify the process. It's worthwhile to use this type of model because it finds an adequate hyperparameter combination in relatively few iterations.

Bayesian optimization is helpful when the objective function is costly in computing resources and time. A drawback compared to grid search or random search is that we must compute Bayesian optimization sequentially (where the

next iteration depends on the previous one), so it doesn't allow distributed processing. So, Bayesian optimization takes longer yet uses fewer computational resources.

Ref : https://www.anyscale.com/blog/what-is-hyperparameter-tuning

11. What issues can occur if we have a large learning rate in Gradient Descent?

12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

13. Differentiate between Adaboost and Gradient Boosting.

Ans:

Features	**Gradient boosting**	**Adaboost**
Model	It identifies complex observations by huge residuals calculated in prior iterations	The shift is made by up-weighting the observations that are miscalculated prior
Trees	The trees with week learners are constructed using a greedy algorithm based on split points and purity scores. The trees are grown deeper with eight to thirty-two terminal nodes. The week learners should stay a week in terms of nodes, layers, leaf nodes, and splits	The trees are called decision stumps.
Classifier	The classifiers are weighted precisely and their prediction capacity is constrained to learning rate and increasing accuracy	Every classifier has different weight assumptions to its final prediction that depend on the performance.
Prediction	It develops a tree with help of previous classifier residuals by capturing variances in data. The final prediction depends on the maximum vote of the week learners and is weighted by its accuracy.	It gives values to classifiers by observing determined variance with data. Here all the week learners possess equal weight and it is usually fixed as the rate for learning which is too minimum in magnitude.

Short-comings	Here, the gradients themselves identify the shortcomings.	Maximum weighted data points are used to identify the shortcomings.
Loss value	Gradient boosting cut down the error components to provide clear explanations and its concepts are easier to adapt and understand	The exponential loss provides maximum weights for the samples which are fitted in worse conditions.
Applications	This method trains the learners and depends on reducing the loss functions of that week learner by training the residues of the model	Its focus on training the prior miscalculated observations and it alters the distribution of the dataset to enhance the weight on sample values which are hard for classification

Ref: https://www.educba.com/gradient-boosting-vs-adaboost/

14. What is bias-variance trade off in machine learning?

Ans:

Whenever we discuss model prediction, it's important to understand prediction errors (bias and variance). There is a tradeoff between a model's ability to minimize bias and variance. Gaining a proper understanding of these errors would help us not only to build accurate models but also to avoid the mistake of overfitting and underfitting.

So let's start with the basics and see how they make difference to our machine learning Models.

What is bias?

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

What is variance?

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't

seen before. As a result, such models perform very well on training data but has high error rates on test data.

Mathematically

Let the variable we are trying to predict as Y and other covariates as X. We assume there is a relationship between the two such that

Y=f(X) + e

Where e is the error term and it's normally distributed with a mean of 0.

We will make a model f^(X) of f(X) using linear regression or any other modeling technique.

So the expected squared error at a point x is

$$Err(x) = E\left[(Y - \hat{f}\,(x))^2\right]$$

The Err(x) can be further decomposed as

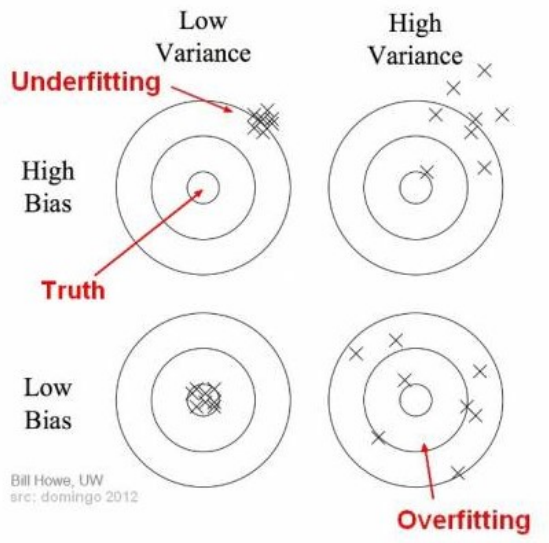
$$Err(x) = \left(E[\hat{f}\,(x)] - f(x)\right)^2 + E\left[\left(\hat{f}\,(x) - E[\hat{f}\,(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Err(x) is the sum of Bias², variance and the irreducible error.

Irreducible error is the error that can't be reduced by creating good models. It is a measure of the amount of noise in our data. Here it is important to understand that no matter how good we make our model, our data will have certain amount of noise or irreducible error that can not be removed.

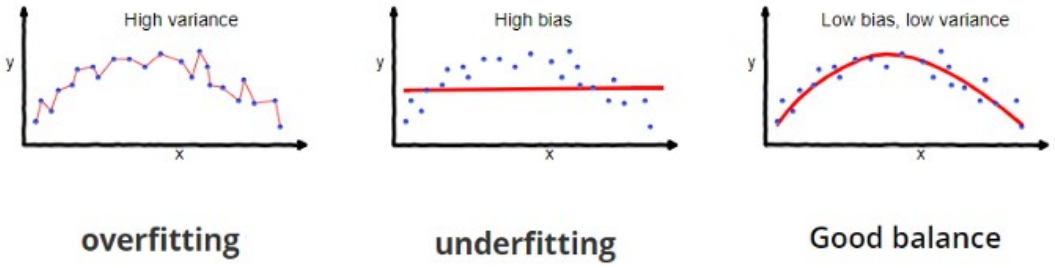
Bias and variance using bulls-eye diagram



In the above diagram, center of the target is a model that perfectly predicts correct values. As we move away from the bulls-eye our predictions become get worse and worse. We can repeat our process of model building to get separate hits on the target.

In supervised learning, **underfitting** happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kind of models are very simple to capture the complex patterns in data like Linear and logistic regression.

In supervised learning, **overfitting** happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.



Why is Bias Variance Tradeoff?

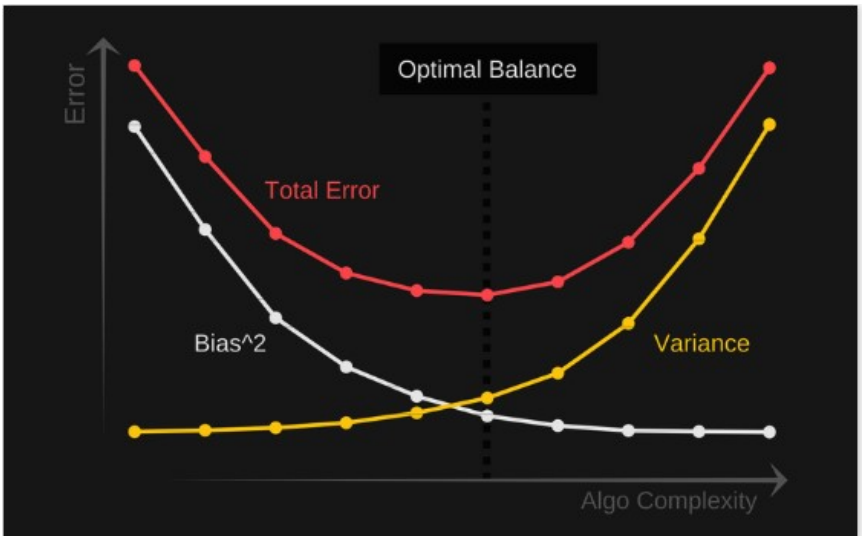
If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.

This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

Total Error

To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.

Total Error = Bias^2 + Variance + Irreducible Error



An optimal balance of bias and variance would never overfit or underfit the model.

Therefore understanding bias and variance is critical for understanding the behavior of prediction models.

Ref : https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.