Worksheet Set – 2

Question 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?

Answer:- R-Squared and Residual sum of squares (RSS) are two different measures of goodness of fit for a regression model.

They are related, but they have different interpretations and properties.

- **RSS :-** Residual sum of squares (RSS) is the sum of squares of error terms and this is an absolute number.

  This number can vary greatly based on the number of data points. RSS will increase if the data points are more and the value will decrease if the data points are less.

  RSS = (Y1 – Ypred)^2 + (Y2 – Ypred)^2+ ----- +(Yn – Ypred)^2

- **R-Squared:-** R-Squared is a measure of variance for dependent variables.
  That is variance in the output that is explained by the small change in input.

  R^2 = 1- RSS/TSS

  The value of R-Squared is always between 0 (0%) and 1 (100%). The bigger the value better the fit.

R-Squared is the proportion of the variance in the response variable that is explained by the regression model.

It ranges from 0 to 1, where 0 means no fit and 1 means perfect fit.

R-Squared is useful for comparing different models or variables, as it adjusts for the scale of the response variable.

However, R-Squared does not tell us how well the model predicts new data, and it can be artificially inflated by adding more variables to the model.

RSS is the sum of the squared differences between the observed and predicted values of the response variable.

It measures the absolute amount of error in the model.

RSS depends on the scale of the response variable and the number of data points.

It is useful for finding the best-fitting model among a set of candidates, as it minimizes the error.

However, RSS does not tell us how much of the variation in the response variable is captured by the model, and it can be arbitrarily large or small depending on the data.

R-Squared and RSS are both measures of goodness of fit, but they answer the different questions.

R-Squared tells us how well the model explains the data,

While RSS tells us how well the model fits the data.

Neither of them is better or worse than the other, but they should be used appropriately for the purpose of the analysis.

Question 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.

Answer:- TSS, ESS And RSS are statistical measures of variability in regression analysis.

They indicate how much the dependent variable deviates from the mean and the predicted values.

- **TSS** is the total sum of squares, which is the sum of squared differences between the observed dependent variable and the overall mean.
  It measures the total variability of the data.

- **ESS** is the explained sum of squares, which is the sum of squared differences between the predicted value and the mean of the dependent variable.
  It measures how well the regression model fits the data.

- **RSS** is the residual sum of squares, which is the sum of squared differences between the observed and predicted values.
  It measures the unexplained variability of the data.

The equation relating these three metrics is:

TSS = ESS + RSS

That formula just says that the total variability equals what is explained by a regression model plus what is left unexplained (the residual).

This means that the total variability of the data can be decomposed into the variability explained by the regression model and the variability that remains unexplained by the model.

The best fit for a regression line is the one that minimizes the RSS, which is called the **Ordinary Least Square (OLS).**

Question 3:- What is the need of regularization in machine learning?

Answer:- Regularizations becomes necessary in machine learning when the model begins to overfit or underfit.

Overfitting occurs when the model performs well with the training data but does not perform well with the test data.

Regularizations is a technique to prevent the model from overfitting by adding extra information to it.

It is a regression that diverts or regularizes the coefficient estimates towards zero.

It reduces flexibility and discourages learning in a model to avoid the risk of overfitting.

Overfitting:- When a model performs well on the training data and does not perform well on the testing data, then the model is said to have high generalization error.

In other words, in such a scenario, the model has low bias and high variance and is too complex, This is called **Overfitting.**

So, how do we prevent this overfitting? How to ensure that the model predicts well both on the training and the testing data? One way to prevent overfitting is **Regularization.**

There is a principle called Occam's Razor, which states : "When faced with two equally good hypothesis, always choose the simpler."

Regularization is an application of Occam's Razor.

It is one of the key concepts in machine learning as it helps choose a simple model rather than the complex one.

We want our model to perform well both on the train and the new unseen data, meaning the model must have the ability to be generalized.

Generalization error is "a measure of how accurately an algorithm can predict outcome values for previously unseen data."

Regularization refers to the modifications that can be made a learning algorithm that helps to reduce this generalization error and not the training error.

It reduces by ignoring the less important features.

It also helps prevent overfitting, making the model more robust and decreasing the complexity of a model.

The regularization techniques in machine learning are:

- **Lasso regression**: having the L1 norm
- **Ridge regression**: with the L2 norm
- **Elastic net regression**: it is a combination of Ridge and Lasso regression.

The regularization in machine learning is used in following Scenarios:

- **Ridge regression** is used when it is important to consider all the independent variables in the model or when many interactions are present.
  That is where collinearity or codependency is present amongst the variables.

- **Lasso regression** is applied when there are many predictors available and would want the model to make feature selection as well for us.

- When many variables are present, and we can't determine whether to use Ridge or Lasso regression, then the **Elastic-Net Regression** is used.

Question 4:- What is Gini–impurity index?

Answer:- Gini-impurity index is a measure of the impurity or uncertainty of a dataset.

It is used in decision tree learning to determine the optimal split for a given node.

The Gini impurity of a dataset is a number between 0 and 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.

The Gini index is the additional approach to dividing a decision tree.

Purity and impurity in a junction are the primary focus of the Entropy and information Gain framework.

The Gini index, also known as Impurity, calculates the likelihood that somehow a randomly picked instance would be erroneously cataloged.

Gini impurity index is a measure of how likely a randomly chosen element from a set would be incorrectly labelled if it was randomly labelled according to the distributions of labels in the subset.

It is used in decision tree algorithms to find the best feature to split the data at each node.

The lower the Gini impurity, the more homogeneous the subset is.

The formula for Gini impurity is:

Gini(D) = 1 – k ∑ i=1 (Pi)^2

Where D is the dataset, k is the number of classes, and Pi is the probability of an element belonging to class i.

When training a decision tree, the attribute that provides the smallest Gini impurity is chosen to split the node.

If a dataset D is split on an attribute A into two subsets D1 and D2 with sizes n1 and n2, respectively, the Gini impurity can be defined as:

Gini A (D) = n1/n Gini(D1) + n2/n Gini(D2)

Where n is the total number of samples in D.

For example: If a node has 10 elements, 6 of which belong to class A and 4 of which belong to class B, the Gini impurity of that node is:

Gini(D) = 1 – (6/10)^2 – (4/10)^2 = 0.48

The Gini impurity of that node is 0.48

Question 5:- Are unregularized decision-trees prone to overfitting? If yes, why?

Answer:- Decision tress are a type of supervised machine learning algorithm that can be used for both classification and regression tasks.

They split the data into subsets based on the values of the features , until a stopping criterion is met.

However, decision tress can suffer from overfitting, which means that they fit the training data too closely and do not generalize well to new or unseen data.

Decision Tress are prone to over-fitting.

Overfitting occurs when the tree becomes to complex and captures the noise in the training data, rather than the underlying pattern.

Allowing a decision tree to split to a granular degree makes it prone to learning every point extremely well, to the point of perfect classification, i.e. **Overfitting.**

Pruning can help to reduce the complexity and avoid overfitting.

One needs to pay special attention to the parameters of the algorithms to understand how each of them could contribute to overfitting.

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.

To prevent overfitting, decision trees can use techniques such as pruning, which removes parts of the tree that do not contribute to the accuracy or information gain.

Pruning can be done before or after the tree is fully grown, and it can reduce the variance and complexity of the tree.

Another way to avoid overfitting is to use ensemble methods, such as random forest, which combine multiple decision trees and average their predictions.

This can reduce the bias and variance of the individual trees and improve the overall performance.

Question 6:- What is an ensemble technique in machine learning?

Answer:- Ensemble techniques in machine learning are methods that created multiple models and then combine them to produce improved results.

Ensemble techniques can help to reduce the error caused by noise, variance, and bias in the data.

Some common ensemble techniques are **boosting, bagging, voting, and stacking.**

Boosting and bagging are sequential and parallel methods that use multiple models of the same type, while voting and stacking are methods that use multiple models of different types.

Ensemble methods allow us to take a sample of Decision Tress into account, calculate which feature to use or questions to ask at each split, and make a final prediction based on the aggregated results of the sampled Decision Trees.

Ensemble methods are a set of machine learning techniques that combine multiple models to improve the overall performance of the model.

The idea behind the ensemble techniques is to train multiple models on the same dataset and then combine their predictions to produce a final prediction.

This approach can help reduce overfitting and improve the accuracy of the model.

There are several types of ensemble methods, including bagging, boosting, voting and stacking.

Ensemble techniques can be classified on the basis of how we are arranging different or same machine learning models.

1) In **Voting** ensemble technique, we take different machine learning algorithms and feed them the same training data to train.
   In regression problems the final outcome will be the mean of all the outcomes predicted by different machine learning models.
   In the classification problem, the voting ensemble technique is further segregated into two parts:

- Hard voting
- Soft voting

In the hard voting classifier, we use ‘Majority count’ which means we go for that outcome that is predicted by most of the machine learning models.

In soft voting classifier, we take the mean of the probability of the outcomes predicted by different machine learning models and then make the final prediction.

2) **Bagging** is a technique that involves training multiple models on different subsets of the training data and then combining their predictions.

This approach can help reduce overfitting and improve the accuracy of the model **Random forest** is an example of the bagging algorithm that uses Decision trees as the base model.

3) **Boosting** is another ensemble method that involves training multiple models sequentially, with each model trying to correct the errors of the previous model.

   This approach can help improve the accuracy of the model by focusing on the examples that are difficult to classify **AdaBoost** is an example of a boosting algorithm that uses decision trees as the base model.

4) **Stacking** is a technique that involves training multiple models and then using another model to combine their predictions.
   This approach can help improve the accuracy of the model by combining the strengths of different models.
   Stacking can be used with any type of model, including decision trees, neural networks, and support vector machines.

   Stacking ensemble faces an overfitting problem because we are using the same training dataset to train the base models and also using the prediction of the same training dataset to train the meta-model.
   To solve this problem stacking ensemble comes up with two methods:

- **Blending**
- **K-fold**

1) **Blending:-** in this method, we divide the training dataset into two parts.
   The first part of the training dataset will be used to train the base models then the second part of the training dataset is used by the base models to predict the outcome which is further used by the meta-model.
2) **K-fold:-** in this method, we divide the training dataset into k parts/folds, then k-1 parts of the training dataset are used to train the base models and one part which is left is used by the base models to predict the outcome which is further used by the meta-model.

Question 7:- What is the difference between Bagging and Boosting techniques?

Answer:- Bagging and Boosting are two types of ensemble techniques that combine multiple models to improve the accuracy and stability of predictions.

Bagging uses bootstrap sampling to create different subsets of data and trains the same type of model on each subset.

Boosting uses weighted sampling to create different subsets of data and trains different types of models sequentially, adjusting the weights based on the previous errors.

Bagging reduces variance and overfitting, while Boosting reduces bias and underfitting.

1) **Bagging:-** The bagging technique in machine learning is also known as Bootstrap aggregation.
   it is a technique foe lowering the prediction model's variance.
   Regarding bagging and boosting, the former is a parallel strategy that trains several learners simultaneously by fitting them independently of one another.
   Bagging leverages the dataset to produce more accurate data for training.
   This is accomplished when the original dataset extracts replacement sampling for subsequent usage.
   When sampling with replacement, every new training data set may experience repetition in certain observations.
   Every component of Bagging has an equal chance of turning up in a fresh dataset.

Bagging Work:-

- The actual dataset (original one) is divided and categorized into numerous subsets, picking observations with replacements.

- There will be the creation and development of a base model in every subset.

- As the subsets are independent of one another, there will be parallel training conducted for each model.

- The final prediction is derived and made once you integrate the predictions from each and every model.

Several machine learning experts use bagging as a technique to create ML models for the healthcare sector.

Advantages of Bagging:-

- **Enhanced Accuracy:-** Bagging boosts the accuracy and precision of the ML (machine learning) algorithms to ensure statistical classification and regression.

- **Lowers variance:-** It lowers the overfitting and variance to devise a more accurate and precise learning model.

- **Weak learners conversion:-** Parallel processing is the most efficient solution to convert weak learner models into strong learners.

Example:- When comparing bagging Vs. boosting, the former leverages the Random Forest model.

This model includes high-variance decision tree models.

It lets you grow trees by enabling random feature selection.

A random forest comprises numerous random trees.

2) **Boosting:-** The sequential ensemble technique known as ‘’Boosting’’ iteratively modifies the weight of each observation based on the most recent categorization.
   The weight of the observation is increased if it is mistakenly classified.
   It creates robust predictive models and reduce bias error.

Boosting Work:-

- The training dataset creates a subset comprising data points with equivalent weightage.

- For the initial dataset, you’ll create a base model.
  This model can later serve the purpose of making predictions on the whole dataset.

- The actual and predicted values will be used to calculate the determined errors.
  Higher weightage is given to the incorrectly predicted observations.

- While the subsequent model is created or devised, the boosting attempts to rectify the previous model’s error.
  The procedure is repeated for numerous models , and every time, boosting rectifies the errors of the last model.

- In the end, the final model comes out as a strong learner.
  It is then considered the weighted means of every other model.

Advantages of Boosting:-

- **Reduce variance:-** Boosting techniques in machine learning enables a quick solution to the two-classification problem while lowering the variance effectively.

- **Deals with missing data:-** Boosting is beneficial in dealing with missing data. That's because numerous models are connected sequentially to resolve the issue of missing data.

Example:- The AdaBoost leverages the boosting techniques in machine learning, where the model maintenance necessitates less than 50% error.

Here a single learner can either be discarded or kept via boosting. If not, the steps are repeated until a strong learner is achieved.

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

Answer:- OOB (out_of_bag) errors are an estimate of the performance of a random forest classifier or regressor on unseen data.

In scikit-learn, the OOB error can be obtained using the OOB_score_attribute of the random forest classifier or regressor.

The OOB error is computed using the samples that were not included in the training of the individual trees.

Out_of_bag error is a performance metric for random forests, which one ensemble machine learning models composed of multiple decision trees.

Out_of_bag error is calculated using the samples that are not used in the training of the individual trees, and provides an unbiased estimate of the model's accuracy on unseen data.

- Out_of_bag error can be obtained using the oob_score_attribute of the random forest classifier or regressor in scikit_learn.

- Out_of_bag error can be used to select the optimal number of trees (n-estimators) for the random forest, by finding the point where the error stabilizes.

- Out_of_bag error is different from the validation error, which is the performance of the model on a separate validation dataset that the model has not seen during training.

Question 9:- What is K-fold cross-validation?

Answer:- 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.

k-fold cross validation is a procedure to evaluate machine learning models on a limited data sample.

A given dataset is split into k equal sized sections/folds.

Each fold is used as a testing set once, while the remaining folds as used as training data.

The model performance is measured by the average score on each fold.

k-fold cross validation is a technique for evaluating the performance of a machine learning model on unseen data.

It involves splitting the data into k subsets or folds, and using the one fold as a validation set and the rest as the training set.

The model is trained and tested k times, each time using a different fold as the validation set.

The average of the performance metrics from each fold is used to estimate the model's generalization ability.

Some of the benefits of k-fold cross validation are:-

- It reduces the bias and variance of the model estimation compared to a simple train/test split.

- It makes use of all the data for both training and testing, which is useful when the data is limited.

- It allows for tuning the model's hyperparameters by selecting the value of k that gives the best performance.

Some of the drawbacks of k-fold cross validation are:-

- It can be computationally expensive and time-consuming, especially for large datasets and complex models.

- It can introduce some randomness in the results, depending on how the data is shuffled and partitioned.

- It may not work well for imbalanced or stratified data, where the distribution of the target variable is different across the folds.
  In such cases, stratified k-fold cross validation can be used to preserve the proportion of the target variable in each fold.

Question 10:- What is hyper parameter tuning in machine learning and why it is done?

Answer:- Hyperparameter tuning is the process of choosing a set of optimal hyperparameters for a machine learning algorithm.

A hyperparameter is a model argument whose value is set before the learning process begins.

Hyperparameters tuning can improve the performance of different machine learning algorithms.

Some common strategies for hyperparameter tuning are GridSearchCV and RandomizedSearchCV, which evaluate the model for a range of hyperparameter values.

Hyperparameters are settings that control the learning process of the model, such as the learning rate, the number of neurons in a neural networks, or the kernel size in a support vector machine.

The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task.

Hyperparameter tuning is important because it can improve the model's performance on unseen data, prevent overfitting, and reduce training time.

For example, a poorly calibrated model will have high bias, meaning it is unsuitable for new data.

On the other hand, a well-calibrated model will have low bias and low variance, meaning it will generalize well to new data and be accurate.

Hyperparameter tuning can also help to find the optimal trade-off between complexity and simplicity of a model, which can affect its efficiency and interpretability.

There are different ways of hyperparameter tuning, such as grid search, random search, Bayesian optimization, and gradient-based optimization.

Each method has its own advantages and disadvantages, depending on the problem and the resource available.

Applications of Hyperparameter tuning:-

- **Model Selection:-** Choosing the Right Model Architecture for the task.

- **Regularization Parameter Tuning:-** Controlling model complexity for optimal performance.

- **Feature Preprocessing Optimization:-** Enhancing data quality and model performance.

- **Algorithmic Parameter Tuning:-** Adjusting algorithm-specific parameters for optimal results.

Advantages of Hyperparameter Tuning:-

- Improved model performance
- Reduced overfitting and underfitting
- Enhanced model generalizability
- Optimized resource utilization
- Improved model interpretability

Disadvantages of Hyperparameter tuning:-

- Computational cost
- Time-consuming process
- Risk of overfitting
- No guarantee of optimal performance
- Requires expertise.

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

Answer:- If the learning rate in gradient descent is set too high or large, it can cause the parameter update to "jump over" the ideal minima and subsequent updates will either result in a continued noisy convergence in the general region of the minima, or in more extreme cases may result in divergence from the minima.

This can lead to osculation's around the minimum or in some cases to outright divergence.

The learning rate is a hyperparameter that controls how much to change the model weights in response to the estimated error gradient.

If the learning rate is too large, it can cause the following issues:

- The model may **overshoot** the optimal solution and **oscillate** around the minimum, resulting in unstable training and sub-optimal weights.

- The model may **diverge** from the optimal solution and weights may **explode,** resulting in an overflow error.

- The model may **skip** important features or regions of the data, resulting in poor generalization and high bias.

Therefore, it is important to choose an appropriate learning rate that is neither too large nor too small, as it affects the speed and quality of the learning process.

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

Answer:- Logistic regression is used when the dependent variable is binary, while linear regression is used when the dependent variable is continuous.

Logistic regression assumes a linear relationship between the independent variables and the logit function of the dependent variable, which is not suitable for non-linear data.

There are functions in statistics and machine learning Toolbox for fitting nonlinear regression models, but not for fitting nonlinear logistic regression models.

Logistic regression is a type of regression that models the probability of a binary outcome as a function of some predictor variables.

The predictor variables can be either linear or non-linear, depending on how they are transformed or specified in the model.

One way to use logistic regression for classification of non-linear data is to apply a nonlinear transformation to the predictor variables, such as a polynomial, logarithmic, or exponential function.

This can capture the non-linear relationship between the predictor variables and the outcome variable.

For example: If the outcome variable is Y and the predictor variable is X, we can use a logistic regression model of the form:

Log (Y/1 – Y) = B0 + B1x + B2x^2

This model allows for a quadratic relationship between X and Y, which can be useful for data that has a U-shaped or inverted U-shaped pattern.

Another way to use logistic regression for classification of non-linear data is to use a non-linear function of the predictor variables in the model, such as a spline, a neural network, or a kernel function.

These functions can approximate any non-linear relationship between the predictor variables and the outcome variable, without requiring a specific functional form.

For example, if the outcome variable is Y and the predictor variable is X, we can use a logistic regression model of the form:

Log (Y/1 – Y) = F(X, B)

Where F is a non-linear function of X and B is a vector of parameters.

This model can be estimated using various methods, such as maximum likelihood, generalized least squares, or gradient descent.

Logistic regression can be used for classification of non-linear data by either transforming the predictor variables or using a non-linear function of the predictor variables in the model.

However, these methods have some drawbacks, such as increased complexity, overfitting, or computational cost.

Therefore, it is important to evaluate the performance of the model using appropriate criteria, such as accuracy, precision, recall or AUC.

Logistic regression is a type of linear classifier, which means it assumes that the classes can be separated by a linear boundary in the feature space.

However, this assumption may not hold for some non-linear data, where the classes have more complex shapes or interactions.

In such cases, logistic regression may not be able to capture the true relationship between the features and the class labels and may result in poor accuracy or generalization.

To deal with non-linear data, we can either use a different type of classifier, such as a decision tree, a neural network, or a support vector machine, that can learn more flexible and non-linear decision boundaries.

Alternatively, we can transform the features using some non-linear functions, such as polynomials, splines, or kernels, and then apply logistic regression on the transformed features.

This way, we can still use logistic regression as a classifier, but with a non-linear model.

Question 13:- Differentiate between AdaBoost and Gradient Boosting.

Answer:- AdaBoost and Gradient Boost are both boosting algorithms that use weak learners to create a strong classifier. However, they differ in the following aspects:

AdaBoost	Gradient Boosting
• AdaBoost minimises a loss function related to classification error.	• Gradient boost minimises a general and differentiable loss function.
• AdaBoost adjusts the weights of instances at each iteration	• Gradient boost fits the new predictor to the residual errors of the previous predictor
• AdaBoost can use different base estimators	• Gradient boost uses decision trees as the base estimator
• AdaBoost uses a predefined loss function, which is the exponential loss	• Gradient boost can use any differentiable loss function, such as the squared error or the logistic loss
• AdaBoost adapts to the errors of the weak learners	• Gradient boosts adapt to the residuals of the loss function
• AdaBoost assigns more weight to the weak learners that have higher accuracy and less weight to those that have lower accuracy	• Gradient boosting assigns equal weight to all weak learners, and instead of modifying the sample weights, it fits the next weak learner to the residual errors of the previous weak learner
• AdaBoost is more sensitive to outliers and noise, as it tries to fit every point perfectly	• Gradient boosting is more robust, as it minimizes the loss function and can handle missing values
• AdaBoost is faster and easier to implement, as it has fewer hyperparameters to tune	• Gradient boosting is more flexible and powerful, as it can optimize various loss functions and control the complexity of the weak learners
• AdaBoost works well with discrete and categorical data, as it uses an exponential loss function that is suitable for classification.	• Gradient boosting works well with continuous and numerical data, as it can use different loss functions that are suitable for regression.

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

Answer:- The bias-variance trade-off is fundamental concept in machine learning and statistics.

It describes the relationship between a model's complexity, the accuracy of its predictions and how well it can make predictions on previously unseen data that were not used to train the model.

A model with high bias is too simplistic and underfits the data, while a model with high variance is too complex and overfits the data.

It refers to the trade-off between two types of errors that a model can make: **bias error and variance error**.

Bias error is the difference between the average prediction of the model and the true value.

A model with high bias makes strong assumptions about the data and tends to oversimplify the problem, resulting in underfitting.

A model with low bias is more flexible and can capture the complexity of the data, resulting in better accuracy.

Variance error is the variability of the model's predictions for different data points.

A model with high variance is sensitive to small changes in the data and tends to overfit the noise, resulting in poor generalization.

A model with low variance is more stable and consistent, resulting in better performance on new data.

The bias-variance trade-off is a design consideration when training the machine learning model.

A model with high bias makes strong assumptions about the data and tends to underfit it, meaning it cannot capture the true relationship between the variables.

A model with high variance learns too much from the data and tends to overfit it, meaning it cannot adapt well to new data.

The bias-variance trade-off states that it is impossible to simultaneously minimize both types of errors, as reducing one typically increases the other.

Therefore, the goal of machine learning is to find the optimal balance between bias and variance that minimizes the total error.

This can be achieved by using techniques such as regularization, cross-validation, and model selection.

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

Answer:- A kernel function is a way of transforming the input data into a higher dimensional feature space, where a linear classifier can find an optimal decision boundary.

Here are some common kernel functions:

1) **Linear kernel: -** This is the simplest kernel, which computes the dot product between two input vectors.

   It is used when the data is linearly separable, or when the number of features is large.

   The linear kernel is given by:

   K (x, y) = x^t y

2) **Polynomial kernel: -** This kernel allows for curved decision boundaries by raising the dot product to some power.
   It is controlled by three parameters: the degree of the polynomial, the scaling factor, and the offset.

   The polynomial kernel is given by:

   K (x, y) = (yx^t y+ r) ^d

3) **Radial basis function (RBF) kernel: -** This kernel is also known as the Gaussian kernel, and it is the most popular kernel for SVMs.
   It measures the similarity between two input vectors by computing the squared Euclidean distance between them, and then applying an exponential function.
   It is controlled by one parameter, the width of the Gaussian function.

   The RBF kernel is given by:

   K(x, y) = exp (-y ||x – y||^2)