	WORKSHEET 5- machine learning answer
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?
ANS	R-squared and Residual Sum of Squares (RSS) are both measures of goodness of fit in regression models, but they serve different purposes and cannot be directly compared as to which one is better. R-squared (also known as the coefficient of determination) is a measure of the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, with a value of 1 indicating a perfect fit and a value of 0 indicating that the model explains none of the variation in the dependent variable. R-squared is a commonly used measure of goodness of fit because it is easy to interpret and gives an indication of how well the model fits the data as a whole. On the other hand, Residual Sum of Squares (RSS) is a measure of the total amount of variation in the dependent variable that is not explained by the independent variables in the model. It is calculated by summing the squared differences between the predicted values and the actual values of the dependent variable. A smaller RSS indicates a better fit because it means that the model is able to explain more of the variation in the dependent variable. Both R-squared and RSS have their own advantages and limitations, and their usefulness depends on the specific research question and the goals of the analysis. R-squared is useful for assessing how well the model explains the variation in the dependent variable, while RSS is useful for assessing how well the model predicts the dependent variable. In general, a good model should have a high R-squared and a low RSS, but the specific values depend on the context of the analysis. Ultimately, the choice of which measure to use depends on the research question and the goals of the analysis.

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.
	In regression analysis, Total Sum of Squares (TSS), Explained Sum of Squares (ESS), and Residual Sum of Squares (RSS) are measures used to assess the goodness of fit of a model. TSS measures the total variability in the response variable (y) and is calculated as the sum of the squared differences between the observed values and the mean of the response variable. TSS = Σ(y - y_mean)$^{2.}$ ESS measures the variability in the response variable that is explained by the regression model and is calculated as the sum of the squared differences between the predicted values and the mean of the response variable. ESS = Σ(y_pred - y_mean)^2. RSS measures the variability in the response variable that is not explained by the regression model (i.e., the residuals) and is calculated as the sum of the squared differences between the observed values and the predicted values. RSS = Σ(y - y_pred)^2. The relationship between these three metrics is expressed by the following equation: TSS = ESS + RSS In other words, the total variability in the response variable (TSS) can be partitioned into the variability explained by the regression model (ESS) and the variability that is not explained by the model (RSS). The proportion of TSS that is explained by the model is given by the coefficient of determination (R-squared), which is calculated as R^2 = ESS/TSS
3	What is the need of regularization in machine learning?
	Regularization is an important technique in machine learning that is used to prevent over fitting, which is a common problem when training complex models. Over fitting occurs when a model learns to fit the training data too closely, resulting in poor generalization to new, unseen data.

	Regularization works by adding a penalty term to the loss function that the model is trying to optimize. This penalty term encourages the model to have smaller weights, which in turn reduces the complexity of the model and makes it less likely to overfit. There are several types of regularization techniques used in machine learning, including L1 regularization (also known as Lasso regularization), L2 regularization (also known as Ridge regularization), and dropout regularization. L1 regularization adds a penalty term to the loss function that is proportional to the absolute value of the weights of the model, while L2 regularization adds a penalty term that is proportional to the square of the weights. Dropout regularization randomly drops out some units in the network during training to prevent the network from relying too heavily on any one feature.
4	What is Gini–impurity index?
	The Gini impurity index is a measure of the impurity or homogeneity of a set of examples in a binary classification problem. It is commonly used in decision tree algorithms to evaluate the quality of a split. The Gini impurity of a set S is defined as: Gini(S) = 1 - sum(p_i^2) where p_i is the proportion of examples in S that belong to class i. If a set S contains only examples from one class, the Gini impurity is 0. If the set is equally distributed among two classes, the Gini impurity is 0.5, which is the highest possible value. When constructing a decision tree, the Gini impurity index is used to evaluate each potential split in the data. The goal is to find the split that maximally reduces the impurity of the resulting subsets. This process is repeated recursively until a stopping criterion is met, such as when the tree reaches a maximum depth or a minimum number of examples per leaf node. The Gini impurity index is one of several measures of impurity that can be used in decision tree algorithms, including entropy and classification error.

5	Are unregularized decision-trees prone to overfitting? If yes, why?
	Yes, unregularized decision trees are prone to overfitting, especially when they are trained on complex or noisy datasets. This is because decision trees are capable of learning very complex decision boundaries that can perfectly fit the training data, including noise and outliers, even if they are not representative of the underlying true distribution of the data. Overfitting occurs when a model is too complex, and it captures the noise in the training data instead of the underlying patterns. As a result, the model performs well on the training data but poorly on the unseen data. Decision trees can easily become overfit because they can continue to split the data until each leaf node only contains a single sample, which leads to a perfect fit for the training data but does not generalize well to new data. To prevent overfitting in decision trees, various regularization techniques can be used, such as pruning, setting a maximum depth, minimum sample split, or minimum sample leaf size, and using ensemble methods such as random forests. These techniques prevent the tree from becoming too complex and force it to focus on the most informative features and patterns in the data.
6	What is an ensemble technique in machine learning?
	An ensemble technique in machine learning is a method of combining the predictions of multiple models to produce a more accurate and robust prediction. The idea behind ensemble techniques is that by combining the predictions of several models, the errors and biases of individual models can be reduced, resulting in a more accurate and reliable prediction. Ensemble techniques can be used with any type of model, but they are particularly effective with complex models such as decision trees, neural networks, and support vector machines. There are two main types of ensemble techniques: **Bagging (Bootstrap Aggregating):** This technique involves training multiple instances of the same model on different subsets of the training data. Each model is trained on a random sample of the data with replacement, and the predictions of all

	models are combined to produce a final prediction. Bagging can help reduce the variance of the model and improve its generalization performance. **Boosting**: This technique involves training multiple models sequentially, with each model attempting to correct the errors of the previous model. Boosting can be used with a variety of models, including decision trees and neural networks. Popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost. Ensemble techniques can also be used with other methods such as random forests and stacking, which combine multiple models in more complex ways to further improve performance. Ensemble techniques are widely used in machine learning competitions and real-world applications due to their ability to improve the accuracy and robustness of predictions.
7	What is the difference between Bagging and Boosting techniques?
	Bagging (Bootstrap Aggregating) and Boosting are two ensemble techniques in machine learning that are used to improve the performance of predictive models. Although they share some similarities, there are some key differences between the two techniques: **Training Approach**: In bagging, multiple instances of the same model are trained on different subsets of the training data with replacement. The predictions of all models are then combined to produce a final prediction. In boosting, multiple models are trained sequentially, with each model attempting to correct the errors of the previous model. **Data Sampling**: In bagging, each model is trained on a random sample of the data with replacement. In boosting, each model is trained on the full dataset or a weighted version of the dataset that gives more importance to the misclassified samples from previous models. **Model Weighting:** In bagging, all models are given equal weight when making predictions. In boosting, models that perform better on the training data are given higher weight and have more influence on the final prediction.

	Error Correction: In bagging, the final prediction is made by averaging the predictions of all models, which can help reduce the variance of the model and improve its generalization performance. In boosting, each model attempts to correct the errors of the previous model, which can help reduce the bias of the model and improve its accuracy. Overall, the main difference between bagging and boosting is that bagging aims to reduce the variance of the model by averaging the predictions of multiple models, while boosting aims to reduce the bias of the model by sequentially training models that correct the errors of the previous models
8	What is out-of-bag error in random forests?
	In random forests, each decision tree is trained on a random subset of the training data, which is called the bootstrap sample. This means that some data points are not included in the bootstrap sample for a particular tree, which is known as out-of-bag (OOB) data. The OOB error in random forests is an estimate of the error rate of the model on new, unseen data. It is calculated by evaluating the predictions of each decision tree in the forest on their corresponding OOB data points. Since the OOB data points were not used to train the tree, they provide an unbiased estimate of the model's performance on new data. The OOB error is useful because it provides an estimate of the generalization error of the model without the need for a separate validation set. It can also be used to tune the hyperparameters of the model, such as the number of trees in the forest or the maximum depth of the trees, by selecting the values that minimize the OOB error. Overall, the OOB error is a useful tool for evaluating the performance of random forests and optimizing their hyperparameters.
9	What is K-fold cross-validation?
	K-fold cross-validation is a technique for evaluating the performance of a machine learning model on a limited amount of data. The basic idea is to split the available data

	into K equally sized subsets or "folds", use K-1 folds for training the model, and evaluate its performance on the remaining fold. This process is repeated K times, with each fold used exactly once for testing the model. The steps for performing K-fold cross-validation are as follows: Split the available data into K equally sized subsets. For each subset: Train the model on the K-1 subsets. Evaluate the model on the remaining subset. Record the evaluation metric (e.g., accuracy, precision, recall, F1 score) for that subset. Calculate the average evaluation metric over all K subsets. K-fold cross-validation can be used for a variety of machine learning tasks, including classification, regression, and clustering. It helps to reduce the variance of the evaluation metric by using multiple independent subsets for testing the model. It also helps to ensure that the model generalizes well to new data by evaluating its performance on multiple independent subsets. Common choices for the value of K are 5 and 10, but other values can be used depending on the size of the available data and the computational resources available. Larger values of K can result in lower bias but higher variance, while smaller values of K can result in higher bias but lower variance. Overall, K-fold cross-validation is a powerful tool for evaluating the performance of machine learning models and selecting the best model for a particular task.
10	**What is hyper parameter tuning in machine learning and why it is done?**
	In machine learning, hyperparameter tuning refers to the process of selecting the optimal values for the hyperparameters of a model. Hyperparameters are parameters that are set before training the model and cannot be learned from the data. Examples of hyperparameters include the learning rate of an optimizer, the regularization strength, the number of hidden layers in a neural network, or the number of trees in a random forest.

	Hyperparameter tuning is important because the choice of hyperparameters can have a significant impact on the performance of the model. For example, a high learning rate can cause the optimizer to overshoot the minimum of the loss function and converge to a suboptimal solution, while a low learning rate can slow down the convergence of the optimizer and result in a longer training time. There are different approaches to hyperparameter tuning, including manual tuning, grid search, and random search. In manual tuning, the hyperparameters are selected based on the prior knowledge or intuition of the practitioner. In grid search, a predefined set of values is tested for each hyperparameter, and the combination that results in the best performance is selected. In random search, a random set of values is tested for each hyperparameter, and the combination that results in the best performance is selected Overall, hyperparameter tuning is an important step in the machine learning pipeline to optimize the performance of the model and improve its ability to generalize to new data.
11	**What issues can occur if we have a large learning rate in Gradient Descent?**
	If we have a large learning rate in gradient descent, it can lead to several issues, including: **Divergence:** A large learning rate can cause the optimization algorithm to overshoot the minimum of the cost function and diverge, meaning the cost function increases rather than decreases over time. **Oscillations:** A large learning rate can cause the optimization algorithm to oscillate around the minimum of the cost function, resulting in slow convergence or even no convergence at all. **Instability:** A large learning rate can make the optimization process unstable and sensitive to small changes in the input data, making the model less robust to noise and outliers. **Poor generalization**: A large learning rate can cause the model to fit the training data too closely, resulting in overfitting and poor generalization to new data. To prevent these issues, it is important to choose an appropriate learning rate for the optimization algorithm. This can be done by monitoring the cost function during training and adjusting the learning rate if necessary. Several techniques, such as

	learning rate schedules, momentum, and adaptive learning rate methods, have been developed to automatically adjust the learning rate during training and improve the convergence and generalization of the model.
12	**Can we use Logistic Regression for classification of Non-Linear Data? If not, why**
	Logistic regression is a linear classification algorithm, which means that it can only be used to classify linearly separable data. Linearly separable data is data that can be separated by a linear boundary, such as a line or a plane, into two or more classes. If the data is not linearly separable, logistic regression may not be the best algorithm to use for classification. However, there are ways to transform the data so that it becomes linearly separable, such as by adding polynomial features or using non-linear transformations. Alternatively, non-linear classification algorithms, such as decision trees, random forests, support vector machines, or neural networks, can be used for classification of non-linear data. These algorithms can learn complex non-linear decision boundaries and are better suited for classification tasks where the data is not linearly separable. In summary, while logistic regression can only be used for linearly separable data, there are ways to transform the data so that it becomes linearly separable, or other non-linear classification algorithms can be used instead.
13	Differentiate between Adaboost and Gradient Boosting.
	Adaboost (Adaptive Boosting) and Gradient Boosting are both ensemble learning techniques that combine multiple weak learners to form a strong learner. However, there are several differences between the two methods: Weight updating: Adaboost assigns weights to each training example and updates them based on the misclassification rate of the previous weak learner. The next weak learner then focuses on the misclassified examples to improve the overall performance. In contrast, Gradient Boosting updates the weights of the residual errors of the previous weak learner, allowing the next weak learner to focus on the remaining errors.

	Loss function: Adaboost uses the exponential loss function, which assigns a higher weight to misclassified examples and penalizes them more strongly. Gradient Boosting can use different loss functions, such as mean squared error, cross-entropy, or absolute error, depending on the problem at hand. Weak learners: Adaboost uses a variety of weak learners, such as decision stumps or decision trees with one level, that are sequentially trained on the weighted data. Gradient Boosting typically uses decision trees as weak learners, but they are trained in a greedy and recursive manner, where each tree tries to minimize the loss function. Parallelization: Adaboost can be parallelized easily, as the training of each weak learner is independent of the others. Gradient Boosting is harder to parallelize, as the trees depend on the previous trees and the residual errors, but there are techniques such as parallelizing the training of each tree or using a subset of features at each split. In summary, Adaboost and Gradient Boosting are both powerful ensemble learning techniques, but they differ in the way the weights are updated, the loss function used, the type of weak learners employed, and the ease of parallelization. The choice between the two methods depends on the problem at hand and the characteristics of the data
14	What is bias-variance trade off in machine learning?
	The bias-variance tradeoff is a fundamental concept in machine learning that refers to the relationship between the complexity of a model and its ability to generalize to new, unseen data. Bias refers to the difference between the expected or average predictions of a model and the true values of the target variable. High bias occurs when a model is too simple or has too few features, which can cause it to underfit the data and make inaccurate predictions on both the training and test sets.

	Variance, on the other hand, refers to the variability of the model's predictions across different training sets. High variance occurs when a model is too complex or has too many features, which can cause it to overfit the data and make accurate predictions on the training set but poor predictions on the test set. The bias-variance tradeoff arises because increasing the complexity of a model can reduce its bias but increase its variance, and vice versa. The goal of machine learning is to find the right balance between bias and variance that leads to the best generalization performance on new, unseen data. One way to achieve this balance is through regularization, which can reduce the variance of a model by penalizing large weights or complex features. Another way is to use ensemble methods, such as bagging or boosting, which combine multiple models with different biases and variances to reduce the overall error and improve generalization performance. In summary, the bias-variance tradeoff is a key concept in machine learning that highlights the importance of balancing the simplicity and complexity of a model to achieve optimal performance on new, unseen data.
15	Give short description each of Linear, RBF, Polynomial kernels used in SVM.
	SVM (Support Vector Machines) is a popular machine learning algorithm that is often used for classification and regression tasks. SVMs use kernel functions to transform the input data into a higher-dimensional feature space, where linearly separable boundaries can be more easily identified. Here are short descriptions of three common types of kernels used in SVM: Linear kernel: A linear kernel simply computes the dot product between two input vectors, without applying any transformation to the data. This kernel is often used when the data is already linearly separable or when the number of features is very large compared to the number of training examples.

	RBF (Radial Basis Function) kernel: The RBF kernel is a non-linear kernel that maps the input data to an infinite-dimensional feature space. It uses a Gaussian function to measure the similarity between two input vectors, based on their Euclidean distance in the original feature space. The RBF kernel is popular because it can capture complex non-linear relationships between the input variables and the target variable. Polynomial kernel: The polynomial kernel is another non-linear kernel that maps the input data to a higher-dimensional feature space. It computes the dot product between two input vectors after raising them to a certain power, specified by the degree of the polynomial. The polynomial kernel is useful when the decision boundary between classes is curved or when there are interactions between the input variables. In summary, the linear, RBF, and polynomial kernels are commonly used in SVM to transform the input data into a higher-dimensional feature space, where a linearly separable boundary can be identified. The choice of kernel depends on the problem at hand and the characteristics of the data.