**Assignment-4**

**Q.1) Define decision tree and Explain Decision Tree Classification use case with following example.**

* **Decision Tree:**  Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

|  |
| --- |
| Decision Trees modified |

* **Decision Tree Classification use case :** There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

1. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
2. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

**Q.2) Define**

**a) Entropy :** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where, S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

**b) Gini Impurity:** Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index. It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

* Gini index can be calculated using the below formula: **Gini Index= 1- ∑jPj2**

**c) Information Gain:** Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class.

* According to the value of information gain, we split the node and build the decision tree. A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.
* It can be calculated using the below formula: **Information Gain= Entropy(S)- [(Weighted Avg) \*Entropy(each feature)**

**Q.3) Describe Decision Tree Regression use case with following example.**

* **Decision tree regression** is a type of supervised learning algorithm that can be used to predict a continuous target variable. One common use case for decision tree regression is in the field of salary prediction, where the goal is to predict an individual's salary based on their years of experience.

|  |  |  |
| --- | --- | --- |
| **Experience (In Years)** | **Gap** | **Salary(In Thousand)** |
| **2** | **Yes** | **40** |
| **2.5** | **Yes** | **42** |
| **3** | **No** | **52** |
| **4** | **No** | **60** |
| **4.5** | **Yes** | **56** |

* **An example** of this would be using a dataset that contains information on the years of experience and corresponding salary for a number of individuals. The decision tree algorithm would use this data to build a model that can predict an individual's salary based on their years of experience.
* The algorithm works by recursively splitting the data into subsets based on the values of the input features. For example, in the case of years of experience and salary, the algorithm would start by selecting the "years of experience" feature and split the data into subsets based on whether an individual has more or less than a certain number of years of experience.
* This process is then repeated for each subset, with the algorithm selecting the feature that results in the greatest decrease in the impurity of the target variable.
* Once the tree is built, it can be used to make predictions for new data by traversing the tree from the root node to a leaf node, where the predicted value is the mean target value of the training instances in that leaf.
* Decision Tree Regression is a supervised learning algorithm that can be used to predict a continuous target variable. One example use case is salary prediction based on years of experience.
* The algorithm works by recursively splitting the data into subsets based on the values of the input features and making predictions by traversing the tree from the root node to a leaf node.

**Q.4) Enlist Advantages and Disadvantages of decision tree.**

|  |  |  |
| --- | --- | --- |
|  | **Advantages** | **Disadvantages** |
| **1** | Highly intuitive and easy to understand | Overfitting i.e. it is a high variance algorithm. This means that it can easily overfit because it has no Inherent mechanism to stop, thereby creating complex decision rules |
| **2** | Less number of data preparation steps unlike other machine learning algorithms | A decision tree can be highly time-consuming in its training phase, and this problem can be exaggerated if there are multiple continuous Independent variables |
| **3** | It is a non-parametric algorithm i.e. it does not require lot of assumptions | Optimization i.e, at every level, the decision tree algorithm looks for the pure node and doesn't consider how the recent decision will affect the next few stages of splitting |
| **4** | Highly versatile algorithm and can perform multiple roles apart from the standard predictions | Decision trees are unstable Le these are high variance models, and some changes in the data can dramatically change the predictions produced by the model. |
| **5** | A decision tree does not require scaling of data as well. | Decision tree often involves higher time to train the model. |
| **6** | Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent. | Decision tree training is relatively expensive as the complexity and time has taken are more. |
| **7** | A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders. | The Decision Tree algorithm is inadequate for applying regression and predicting continuous values. |

**Q.5) Calculate Entropy of following datasets.**

1. **Entropy(s)= -Pylog2 (Py)- Pn log2 (Pn)**

**= -2/5 log2(2/5)-3/5log2(3/5)**

**=0.97**

1. **Entropy(s)= -Pylog2 (Py)- Pn log2 (Pn)**

**= -1/5log2 (1/5)-4/5 log2 (4/5)**

**= 0.72**

**Q.6) Calculate Gini Impurity of following datasets.**

**1. Gini Impurity = 1- ∑jPj2**

**G = 1-(4/5 + 9/25) = 0.48**

1. **Gini Impurity = 1- ∑jPj2**

**G = 1-(1/25 + 16/25)**

**G = 0.32**

**Q.7) Explain Over fitting in decision tree with suitable example.**

* Overfitting is an undesirable machine learning behavior that occurs when the machine learning model gives accurate predictions for training data but not for new data. Overfitting occurs when the model cannot generalize and fits too closely to the training dataset instead. Overfitting happens due to several reasons, such as:
* The training data size is too small and does not contain enough data samples to accurately represent all possible input data values.
* The training data contains large amounts of irrelevant information, called noisy data.
* The model trains for too long on a single sample set of data.
* The model complexity is high, so it learns the noise within the training data.
* **Example:** For example, let's say we are trying to predict whether a customer will default on a loan based on their credit history. We use a decision tree to model this problem and train it on a dataset of 1000 customers. The tree ends up becoming very complex, with many branches and leaves that represent very specific subsets of the training data. For example, one leaf might represent customers with a credit score of less than 600 who have been late on a mortgage payment in the last 12 months, while another leaf represents customers with a credit score of 700 or higher who have never been late on any payments.
* While the tree may perform very well on the training data, it is likely to perform poorly on new, unseen data. This is because the tree has learned the noise and random fluctuations in the training data, rather than the underlying patterns and relationships that are important for making accurate predictions.
* To avoid overfitting, one common approach is to prune the tree, which involves removing branches that do not improve the performance of the tree on unseen data. Another approach is to use techniques like bagging and boosting which can improve the generalization of decision tree.
* In bagging, the model is trained on different random subsets of the data, which helps to reduce the noise in the training data. Boosting is an ensemble method where multiple weak models are combined to form a strong model. By using boosting, we can reduce the overfitting of the decision tree.
* In addition, we can also use techniques like cross-validation, regularization and early stopping to prevent overfitting. These techniques can be used to find the optimal complexity of the decision tree, while avoiding overfitting.

**Q.8) Explain Hyper-parameter Tuning using GridSearchCV in decision tree.**

* Hyperparameter tuning is the process of tuning the parameters present as the tuples while we build machine learning models. These parameters are defined by us which can be manipulated according to programmer wish. Machine learning algorithms never learn these parameters.
* These are tuned so that we could get good performance by the model. Hyperparameter tuning aims to find such parameters where the performance of the model is highest or where the model performance is best and the error rate is least.
* **Steps To Follow For Hyper Parameter Tuning using GridSearchCV :**

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.tree import DecisionTreeRegressor  from sklearn.ensemble import RandomForestRegressor  df = pd.read(‘Boston.csv’)  print(df) |

* First, we will import all the required libraries and the dataset and do the basic EDA to understand the data.
* Now we will define the independent and dependent variables y and x respectively. We will then split the dataset into training and testing. After which the training data will be passed to the decision tree regression model & score on testing would be computed.

|  |
| --- |
| y = df['medv']  X = df.drop('medv', axis=1)  from sklearn.model\_selection import train\_test\_split, X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size= .30, random\_state=1)  from sklearn.tree import DecisionTreeRegressor  dtr = DecisionTreeRegressor()  dtr.fir(X\_train,y\_train)  print(dtr.score(X\_test,y\_test)) |

* we will define the library required for grid search followed by defining all the parameters or the combination that we want to test out on the model. We have taken only the four hyperparameters whereas you can define as much as you want. If you increase the number of combinations then time complexity will increase.

|  |
| --- |
| from sklearn.model\_selection import GridSearchCV  param\_grid = { 'bootstrap': [True], 'max\_depth': [5, 10, None], 'max\_features': ['auto', 'log2'], 'n\_estimators': [5, 6, 7, 8, 9, 10, 11, 12, 13, 15]} |

* Now we will define the type of model we want to build a random forest regression model in this case and initialize the GridSearchCV over this model for the above-defined parameters.

|  |
| --- |
| rfr = RandomForestRegressor(random\_state = 1)  g\_search = GridSearchCV(estimator = rfr, param\_grid = param\_grid, cv = 3, n\_jobs = 1, verbose = 0, return\_train\_score=True) |

* We have defined the estimator to be the random forest regression model param\_grid to all the parameters we wanted to check and cross-validation to 3. We will now train this model bypassing the training data and checking for the score on testing data. We can check the best parameter by using the best\_params\_ function that is shown above.

|  |
| --- |
| g\_search.fit(X\_train, y\_train);  print(g\_search.best\_params\_)  print(best\_grid.score(X\_test, y\_test)) |

**Q.9) Describe Bagging process in detail.**

* Bagging(Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Here idea is to create several subsets of data from training sample chosen randomly with replacement. Now, each collection of subset data is used to train their decision trees.
* As a result, we end up with an ensemble of different models. Average of all the predictions from different trees are used which is more robust than a single decision tree.
* Random Forest is an extension over bagging. It takes one extra step where in addition to taking the random subset of data, it also takes the random selection of features rather than using all features to grow trees. When you have many random trees. It’s called Random Forest.
* **Let’s look at the steps taken to implement Random forest:**

**1**. Suppose there are N observations and M features in training data set. First, a sample from training data set is taken randomly with replacement.

**2**. A subset of M features are selected randomly and whichever feature gives the best split is used to split the node iteratively.

**3**. The tree is grown to the largest.

**4**. Above steps are repeated and prediction is given based on the aggregation of predictions from n number of trees.

* **Advantages of using Random Forest technique:**
* Handles higher dimensionality data very well.
* Handles missing values and maintains accuracy for missing data.
* **Disadvantages of using Random Forest technique:**
* Since final prediction is based on the mean predictions from subset trees, it won’t give precise values for the regression model.

**Q.10) Explain Random Forest Classifier with suitable example.**

* **Random Forest Classifier:** Random Forest is an ensemble learning method for classification and regression problems.
* It is a collection of decision trees, where each tree is trained on a random subset of the data. The final prediction is made by averaging the predictions of all the trees in the forest.
* Random Forest is particularly useful for handling high-dimensional and complex data, as it can reduce overfitting and improve the accuracy of the model.
* The basic idea behind Random Forest is to train multiple decision trees on different subsets of the data and then combine their predictions to get a more accurate and stable result.
* Each tree in the forest is trained using a random subset of the data, known as a bootstrap sample.
* In addition, when selecting the features at each split of the tree, a random subset of the features is considered, rather than all of them. This process is called random subspace method.
* One of the key advantages of Random Forest is that it can handle a large number of features and it is less prone to overfitting than a single decision tree.
* Moreover, it reduces variance by averaging the predictions of multiple decision trees and reduces bias by using a random subset of features at each split.
* **For example**, let's say we have a dataset of 1000 observations, with 10 features and a binary outcome (0 or 1). We want to train a Random Forest classifier to predict the outcome. We start by training 10 decision trees, each on a random sample of 900 observations (with replacement) and using a random subset of 7 features at each split. Once all the trees are trained, we make predictions on the remaining 100 observations. For each observation, we get a prediction from each tree, and the final prediction is the majority vote of all the trees.
* Random Forest is widely used in many real-world applications, such as image classification, natural language processing, and medical diagnosis.

**Q.11) Explain Random Forest Regressor with suitable example.**

* **Random Forest Regressor:** Random Forest Regressor is an ensemble learning method for regression problems, similar to Random Forest Classifier.
* It is a collection of decision trees, where each tree is trained on a random subset of the data. The final prediction is made by averaging the predictions of all the trees in the forest. Random Forest Regressor is particularly useful for handling high-dimensional and complex data, as it can reduce overfitting and improve the accuracy of the model.
* The basic idea behind Random Forest Regressor is to train multiple decision trees on different subsets of the data and then combine their predictions to get a more accurate and stable result.
* Each tree in the forest is trained using a random subset of the data, known as a bootstrap sample.
* In addition, when selecting the features at each split of the tree, a random subset of the features is considered, rather than all of them. This process is called random subspace method.
* **For example,** let's say we have a dataset of 1000 observations, with 10 features and a continuous outcome. We want to train a Random Forest regressor to predict the outcome. We start by training 10 decision trees, each on a random sample of 900 observations (with replacement) and using a random subset of 7 features at each split. Once all the trees are trained, we make predictions on the remaining 100 observations. For each observation, we get a prediction from each tree, and the final prediction is the mean of all the predictions.
* One of the key advantages of Random Forest Regressor is that it can handle a large number of features and it is less prone to overfitting than a single decision tree.
* Moreover, it reduces variance by averaging the predictions of multiple decision trees and reduces bias by using a random subset of features at each split.
* Random Forest Regressor is widely used in many real-world applications, such as stock price prediction, weather forecasting, and energy consumption prediction.

**Q.12) Describe Random Forest-BiasVariance Tradeoff in detail.**

* Random Forest is an ensemble learning method that can reduce both bias and variance in the predictions of a model. Bias refers to the error that arises from approximating a real-world problem with a simpler model, while variance refers to the error that arises from small fluctuations in the training data.
* In general, increasing the complexity of a model will decrease bias but increase variance, and decreasing the complexity of a model will increase bias but decrease variance.
* In the case of Random Forest, each tree in the forest is trained on a random subset of the data, and the final prediction is made by averaging the predictions of all the trees. This process reduces both bias and variance in the predictions.
* By averaging the predictions of multiple decision trees, each trained on a different subset of the data, Random Forest reduces the variance of the predictions.
* Additionally, by using a random subset of features at each split and by limiting the maximum depth of each tree, Random Forest reduces the bias of the predictions.
* The Random Forest algorithm reduces bias by averaging the predictions of multiple decision trees, each trained on a different subset of the data and a random subset of features at each split.
* Each tree makes a prediction based on its subset of the data and features, and the ensemble of trees makes a prediction that is less influenced by any single tree. Additionally, by limiting the maximum depth of each tree, Random Forest also reduces bias by avoiding the overfitting caused by deep trees.
* However, as the number of trees in the forest increases, the variance of the predictions will decrease and the bias of the predictions will increase.
* This is because, as the number of trees increases, the model becomes more complex and is able to fit the training data better, but it is also more likely to overfit the data.
* Therefore, there is a trade-off between bias and variance in Random Forest, and it is important to find the right number of trees that strikes a balance between bias and variance.
* In practice, the number of trees in the Random Forest can be selected by monitoring the performance of the model on a validation set, or by using techniques such as cross-validation.
* Additionally, other hyperparameters such as the maximum depth of the tree and the minimum number of samples required to split an internal node can also be adjusted to balance the bias-variance trade-off in the Random Forest.
* Overall, Random Forest is a powerful ensemble learning method that can balance the bias-variance trade-off by averaging the predictions of multiple decision trees, each trained on a different subset of the data, and by using a random subset of features at each split and by limiting the maximum depth of each tree.

**Q.13) Describe Hyper parameter tuning for Random Forest algorithm using GridSearchCv and RandomSearchCV.**

* **Hyper parameter tuning for Random Forest:** Hyperparameter tuning is the process of finding the optimal set of hyperparameters for a machine learning model.
* Hyperparameters are the parameters that are not learned from the data during the training process, but are set prior to training.
* In the case of Random Forest, some of the common hyperparameters include the number of trees in the forest, the maximum depth of each tree, the minimum number of samples required to split an internal node, and the number of features to consider when looking for the best split.
* There are two common methods for tuning hyperparameters in Random Forest: Grid Search and Random Search.

1. **Grid Search** is a method for exhaustively searching over a predefined set of hyperparameters. It involves specifying a range of values for each hyperparameter, and the algorithm will train and evaluate a model for each combination of hyperparameter values.

**For example,** if we want to tune the number of trees and the maximum depth of each tree, we can specify a range of values for each and the algorithm will train and evaluate a model for every possible combination of values. GridSearchCV is an implementation of Grid Search in scikit-learn.

1. **Random Search** is a method for sampling random combinations of hyperparameters from a predefined distribution. It involves specifying a distribution for each hyperparameter, and the algorithm will randomly sample from this distribution to generate a set of hyperparameter values. RandomSearchCV is an implementation of Random Search in scikit-learn.

* Both Grid Search and Random Search can be used to find the optimal set of hyperparameters for a Random Forest model, but Random Search is more efficient and is more likely to find a better set of hyperparameters.
* GridSearchCV will try every possible combination of the hyperparameters, which can be computationally expensive, especially when the search space is large.
* On the other hand, Random Search will only sample a random subset of the possible combinations, which can be more efficient and also have a chance to find a better combination of the hyperparameters.
* It's worth noting that, both techniques are computationally expensive and time-consuming, so it's important to have a good understanding of the problem and the model, and to have a good sense of what the reasonable range of hyperparameters is before starting the search.
* Additionally, it's also important to test the model performance on an independent test set after tuning the hyperparameters, to ensure that the model generalizes well to new unseen data.