**Assignment-5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Outlook** | **Temperature** | **Humidity** | **Windy** | **Play tennis** |
| **0** | **Rainy** | **Hot** | **High** | **False** | **No** |
| **1** | **Rainy** | **Hot** | **High** | **True** | **No** |
| **2** | **Overcast** | **Hot** | **High** | **False** | **Yes** |
| **3** | **Sunny** | **Mild** | **High** | **False** | **Yes** |
| **4** | **Sunny** | **Cool** | **Normal** | **False** | **Yes** |
| **5** | **Sunny** | **Cool** | **Normal** | **True** | **No** |
| **6** | **Overcast** | **Cool** | **Normal** | **True** | **Yes** |
| **7** | **Rainy** | **Mild** | **High** | **False** | **No** |
| **8** | **Rainy** | **Cool** | **Normal** | **False** | **Yes** |
| **9** | **Sunny** | **Mild** | **Normal** | **False** | **Yes** |
| **10** | **Rainy** | **Mild** | **Normal** | **True** | **Yes** |
| **11** | **Overcast** | **Mild** | **High** | **True** | **Yes** |
| **12** | **Overcast** | **Hot** | **Normal** | **False** | **Yes** |
| **13** | **Sunny** | **Mild** | **High** | **True** | **No** |

**Q.1 Explain Naïve Bayes Classifier with following example.**

* Naive Bayes classifier is a probabilistic algorithm that uses Bayes' theorem to make predictions or classifications based on prior knowledge and data. The "naive" part of the name comes from the assumption that all the features in the input data are independent of one another.
* There are three main types of Naive Bayes classifiers: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes.
* Gaussian Naive Bayes is used when the data is continuous and is assumed to be normally distributed.
* Multinomial Naive Bayes is used for discrete data such as text classification.
* Bernoulli Naive Bayes is used for binary data where the features are independent booleans (True/False)
* Naive Bayes classifiers are easy to implement and run efficiently, but the independence assumption may not hold in all cases. However, Naive Bayes is still widely used in practice due to its simplicity and good performance in many applications
* **For example,** using a Naive Bayes classifier to predict whether a tennis match will be played or not based on the weather conditions on that day. The classifier would take in data about the temperature, humidity, wind speed, and whether it's sunny or overcast as input features, and output a prediction of whether the match will be played (yes or no).
* The classifier would be trained on a dataset of past weather conditions and whether the corresponding tennis matches were played or not, and would use Bayes' theorem to calculate the probability of the match being played given the input weather conditions.
* It's important to notice that this "Naive" assumption is not always true, but in many cases, it works well in practice.

**Q.2 How to Handle Zero probabilities in Naïve Bayes Classifier.**

* The Naive Bayes classifier is a probabilistic model that is based on the Bayes theorem.
* It assumes that the features in the data are independent of each other, which allows it to make predictions using the probabilities of each feature.
* However, when some of the features have zero probability, it can cause problems for the classifier.

|  |  |
| --- | --- |
| Class | Count |
| Spam | 500 |
| Non-Spam | 100 |

* One way to handle zero probabilities in the Naive Bayes classifier is to use Laplace smoothing. Laplace smoothing is a technique that adds a small value (called the smoothing parameter) to the numerator of the probability estimate for each feature.
* This has the effect of smoothing out the probabilities and avoiding zero probabilities.
* **Example:-** consider a Naive Bayes classifier that is trained to predict whether an email is spam or not spam based on the presence of certain words in the email. The classifier is trained on a dataset that contains 1000 spam emails and 2000 non-spam emails. The following table shows the number of occurrences of the word "discount" in the emails:
* The probability of the word "discount" given that the email is spam (P(discount|spam)) can be calculated as:

**P(discount|spam) = 500 / 1000 = 0.5**

* The probability of the word "discount" given that the email is not spam (P(discount|non-spam)) can be calculated as:

**P(discount|non-spam) = 100 / 2000 = 0.05**

* Now, suppose that the classifier encounters a new email that contains the word "discount". Without Laplace smoothing, the classifier would assign a probability of zero to the word "discount" given that the email is non-spam (P(discount|non-spam) = 0), which would cause the classifier to incorrectly classify the email as spam.
* To avoid this problem, we can use Laplace smoothing by adding a small value (the smoothing parameter) to the numerator of the probability estimate for each feature. For example, if we set the smoothing parameter to 0.1, the probability of the word "discount" given that the email is non-spam (P(discount|non-spam)) becomes:

**P(discount|non-spam) = (100 + 0.1) / (2000 + 2 \* 0.1) = 0.051**

* This avoids the problem of zero probabilities and allows the classifier to make more accurate predictions.

**Q.3 Define KNN Algorithm. How it works.**

* KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution.
* Lazy algorithm means it does not need any training data points for model generation.
* All training data is used in the testing phase. KNN is called lazy algorithm because it does not learn a discriminative function from the training data but memorizes the training dataset instead.
* KNN algorithm stores all available cases and classifies new cases by a majority vote of its k neighbors.
* The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.
* These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance.
* First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbor.
* At times, choosing K turns out to be a process of trial and error.
* Here are the steps involved in the KNN algorithm:

1. Calculate distance
2. Find nearest neighbors
3. Vote for labels

**Q.4 Write Short Note on Cross Validation in KNN.**

* Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
* The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split. Once the data sample is divided into k groups, the learning algorithm is trained on k-1 of the folds and tested on the remaining one.
* This procedure is then repeated k times with a different fold being used as the test set in each iteration.
* The average performance measure across all k iterations is used as the performance measure of the learning algorithm.
* Cross-validation can be used in the training phase of the KNN algorithm to choose the optimal value for K.
* This is done by training and evaluating the KNN algorithm multiple times on different subsets of the training data using different values of K and selecting the value that gives the best performance on the validation data.
* This process helps to avoid Overfitting and improve the generalization performance of the model.

**Q.5 Explain Support Vector Classifier.**

* Support Vector Classifier (SVC) is a linear model for classification and regression.It is called a support vector classifier because the model derives its name from the support vectors that it uses.
* Support vectors are the data points that are closest to the decision boundary and are used to form the hyperplane that separates the classes.
* The goal of the SVC is to find the hyperplane in an N-dimensional space (N is the number of features) that maximally separates the two classes.
* This is done by finding the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.
* The SVC algorithm can be used for both classification and regression tasks.
* For classification, the model is trained using a set of labeled data points and uses these to predict the class label of new data points.
* For regression, the model is trained using a set of data points and their corresponding continuous target variables and uses these to predict the value of the target variable for new data points.

**Q.6 How To Calculate Support Vector Regressor.**

* Support Vector Regressor (SVR) is a type of Support Vector Machine (SVM) that is used for performing regression tasks. It works by finding the hyperplane in a high-dimensional space that maximally preserve the continuity in the output space.
* The SVR algorithm is trained using a set of data points and their corresponding continuous target variables. The goal is to find the hyperplane that best fits the data and has the maximum margin from the nearest data points. This is done by minimizing the following cost function:

**Loss + C \* (sum(|w|))**

* **Where**:
* **Loss** is the loss function used to measure the error between the predicted values and the true values. The most commonly used loss function for SVR is the epsilon-insensitive loss function, which ignores errors smaller than a certain threshold (epsilon).
* **C** is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the loss.
* **|w|** is the L1-norm of the weight vector, which is used to penalize large coefficients and prevent overfitting.
* **To calculate the SVR**, the algorithm first trains on the input data and finds the optimal weight vector (w) and bias term (b) that minimize the cost function. These parameters are then used to make predictions on new data points using the following equation:

**y = w^T \* x + b**

**Where:**

* y is the predicted value
* x is the input data point
* w is the weight vector
* b is the bias term

**Q.7 Describe Linear and Non Linear SVM.**

* Support Vector Machines (SVMs) are a type of supervised machine learning algorithm that can be used for classification or regression. SVMs are based on the concept of finding a hyperplane that maximally separates the data points of different classes in a high-dimensional space.
* There are two types of SVMs: linear and non-linear.
* **Linear SVM:**
* Linear SVMs are used when the data points are linearly separable, which means they can be separated by a single straight line or hyperplane.
* Linear SVMs try to find the hyperplane that maximally separates the points of different classes and has the maximum margin from the nearest data points of any class.
* The maximum margin hyperplane is the hyperplane that is farthest from the training examples of any class.
* **Non-Linear SVM:**
* Non-linear SVMs are used when the data points are not linearly separable.
* In this case, the SVM algorithm uses the kernel trick to transform the input data into a higher-dimensional space where the data points become linearly separable.
* The kernel function is used to compute the dot product of the two inputs in the higher-dimensional space.
* Some commonly used kernel functions are the polynomial kernel, the Radial Basis Function (RBF) kernel, and the sigmoid kernel.