Introduction to Data Mining

Student performance predictive model

Students:

Negah Sarwari

Ketevan Nikolishvili

Teacher:

Witek ten Hove

Table of Contents

Introduction

The K-Nearest Neighbour (KNN) Model

The k-Nearest Neighbours (kNN) algorithm is a machine learning method utilized for classification and regression tasks (Mitchell, 1997). Its principle is the concept of similarity – objects that are closer in feature space are likely to belong to the same class of have similar values (Cover & Hart, 1967).
kNN is a simple and intuitive algorithm and it says that the similar data points belong to the same class or have similar values. To describe in a simple way, if we take 5 black and 10 white balloons from the box, the 16th balloon will supposedly be white based on the kNN algorithm.

The Naïve Bayes Model (NB)

The Naive Bayes (NB) algorithm is a machine learning method used to classify tasks. The name "naive" comes from the assumption that all features in the dataset are independent of each other. During the training phase, the algorithm analyses the features of the data and computes the probability of each feature occurring given each class label, using Bayes' theorem (Manning, Raghavan, & Schütze, 2008).
In the prediction phase, when there is a new data point, the algorithm calculates the probability of that data point belonging to each class label based on the features it contains. It then selects the class label based on the features. It then selects the class label with the highest probability as the predicted class for the new data point.
The main benefit of the Naive Bayes algorithm is the simplicity and efficiency that makes it useful for text classification tasks. The good example is spam detection where the Naive Bayes algorithm can be used.
In some cases, "naive" assumption of feature independence might not be appropriate performance, especially when there are highly correlated features of complex data. In spite of that, Naive Bayes often solves many classification problems.

Crisp Model

Business understanding

Student performance is impacted by numerous factors, and through this analysis, we aim to explore how various elements such as gender, ethnicity, parental education level, lunch provisions, and test preparation courses influence test scores. By examining these variables, we intend to gain insights into the role of parental background and preparatory activities in shaping students' academic outcomes. This exploration will help us understand the multifaceted nature of educational achievement and guide efforts to enhance student performance effectively.

Data understanding

This dataset comes from Kaggle.com and includes information on 1000 students, organized into 8 columns. It gives a detailed picture of how well students are doing in subjects like math, reading, and writing by showing their scores. Besides these scores, the dataset also looks at different factors that might affect their performance. These factors include the student's gender, their race or ethnicity, how much education their parents got, whether they get a standard or free/reduced lunch (which hints at their economic situation), and if they took a test preparation course. By including both scores and these personal factors, the dataset helps us understand what might help or hinder a student's success in school, offering a chance to see how different conditions and efforts can make a difference in education.
(link of the dataset can be found in the Bibliography)

Data preparation

The K-Nearest Neighbour (KNN) Model

We will now explain the steps we used:

Import pandas as pd:

- **Pandas** is a toolkit for data manipulation and analysis. It's used here to load, prepare, and manipulate the dataset before feeding it into machine learning models.

from sklearn.model_selection import train_test_split:

- This function from the **scikit-learn** library is used to split the dataset into two parts: one for training the machine learning model and the other for testing its performance.

from sklearn.preprocessing import StandardScaler, LabelEncoder:

- **StandardScaler** is utilized to adjust the scale of independent variables or features in the dataset, ensuring a consistent and normalized range across the data. It's important because it can be sensitive to the scale of the input features.
- **LabelEncoder** is a utility tool that aids in standardizing labels, restricting them to values ranging from 0 to n_classes 1.

from sklearn.neighbors import KNeighborsRegressor:

- This line of code brings in the K-Nearest Neighbors regressor from scikit-learn. It's a tool used for predicting continuous values, like numbers, rather than categories. It works by storing examples from the training data and using them to make prediction.

from sklearn.naive_bayes import CategoricalNB:

- This line imports the Categorical Naive Bayes classifier, which is used for sorting things into categories. Naive Bayes assumes that the features are independent of each other when we know the category.

from sklearn.metrics import mean_squared_error:

- This is the performance evaluation metrics.
- **mean_squared_error** is used to measure the average of the squares of the errors, i.e., the average squared difference between the estimated values and the actual value. It's used in regression.

The Naïve Bayes Model

Data preparation:

The data pre-processing begins with data cleaning, where initial inspection and treatment of missing values, incorrect data types, and outlier values take place. Following this, feature selection is performed to eliminate irrelevant features, such as identifiers, and highly correlated features that don't significantly contribute to the predictive ability of the model.

Next, categorization is applied to the **'math_score'** variable, transforming it into math performance categories using the **pd.cut()** function, which facilitates the classification task.

Finally, the dataset is divided into training and testing sets, with 80% of the data allocated for training and 20% for testing. This splitting ensures that the model's

performance is assessed on data it hasn't encountered before, offering a strong indication of its ability to generalize.

We use a **CategoricalNB** classifier because both the features and the target variable are categorical. Then, we train the model on the training set. During training, the model learns how the features relate to each category of math_performance.

Categorizing math_score into Performance Levels
In this step, **math_score is** divided into 'Low', 'Medium', and 'High' performance categories using score ranges. This is accomplished using **the pd.cut()** function, which sets up bins to represent score intervals. This conversion is essential for turning the continuous **math_score** into a categorical target variable, which is crucial for classification purposes.
(One of our data sets was not identified as categorical data so we used **([pd.cut(data['math_score'])** to identify math score as categorical data.)

Feature Selection and Data Splitting
Before this step, we picked out the important features for predicting math_performance and designate them as X_nb, while the target variable is labelled as y_nb. Then, the data is split into training and testing sets, which allowed us to evaluate the model's performance on data that it hasn't seen before.

Modeling

KNN Modeling
Initialzing the KNN regressor
In this step, we create an instance of KNeighborsRegressor with the parameter n_neighbors=5, indicating that the prediction for a new instance will be based on its 5 nearest neighbors in the feature space. The choice of n_neighbors (k) is critical as it influences how the model balances bias and variance. A small 'k' can result in overfitting to the training data (high variance), while a large 'k' might oversmooth, leading to underfitting (high bias).

Training the Model
knn_regressor.fit(X_train_knn_scaled, y_train_knn)
This stage involves training the KNN regressor on the pre-processed and scaled training data. By utilizing the. fit () method, the model learns from the input features (X_train_knn_scaled) and the target variable (y_train_knn). The KNN regressor memorizes the training data to use for making predictions in the future. Pre-scaling the feature before this step ensures that all features have an equal contribution to distance calculations between instances.

Making predictions
y_pred_knn = knn_regressor. predict(X_test_knn_scaled)
With the trained model, we predict math scores for the test set using the. predict () method. The method requires the input features of the test set (X_test_knn_scaled), and

it returns predicted numeric values (y_pred_knn) corresponding to each instance in the test set. The predictions are made by identifying the 'k' nearest neighbors to each test instance in the feature space and then averaging (or sometimes taking the median of) the neighbors' target values. his ensures that no single feature dominates the distance calculations between instances when training the KNN regressor.

Evaluating Model Performance

- **mse_knn = mean_squared_error(y_test_knn, y_pred_knn)**

print(f'KNN MSE: {mse_knn}')

he effectiveness of the KNN regressor is assessed by calculating the Mean Squared Error (MSE) between the predicted math scores and the true math scores in the test set. The MSE serves as a concise metric of the model's predictive accuracy, with smaller MSE values indicating superior performance. Conversely, a higher MSE suggests notable discrepancies between the model's predictions and the actual outcomes, which could indicate problems such as overfitting, underfitting, or the necessity for additional feature refinement.

The Naïve Bayes Model (NB) – modeling

The data is already in place because of the previous steps we did for the analysation so than the following happens:

- **Model training.**

We start by setting up and training a **CategoricalNB classifier** using the training dataset. The. fit () method fine-tunes the model based on the dataset's statistical characteristics, enabling it to make accurate predictions on fresh, unseen data.

- **Model evaluation.**

Following the training phase, the classifier is employed to forecast the target variable for the test dataset. Subsequently, the accuracy score function assesses the model's efficacy by contrasting the predicted labels with the true labels, offering a clear-cut measure of accuracy.

Results

An accuracy of 0.45 means that the NB model correctly classified 45% of the instances in the dataset. Which means that around 45% of the predictions made by the NB model matched the actual labels or classes of the dataset.

Mean Squared Error (MSE) is a measure of the average squared difference between the actual and predicted values in a regression problem (James et al., 2013). It's calculated by taking the average of the squared differences between the predicted and actual values for each data point (Montgomery et al., 2012).
In the case of kNN, the MSE value of 252.09400000000002 means that, on average, the squared difference between the actual values and the predicted values made with the use of kNN model is around 252.09400000000002. In the case of insurance claims, the MSE value is considered as high. Based on the high number of MSE in Knn model, we can say that the model's predictions are less accurate.

Conclusion

Our evaluation indicates that both the Naive Bayes (NB) and K-Nearest Neighbors (KNN) models have significant room for improvement. With the NB model's accuracy hovering around 45%, and the KNN model's predictions showing considerable deviation from actual values, it's clear that adjustments are necessary. Enhancing these models could involve exploring additional data features, fine-tuning model parameters, or even exploring alternative modeling techniques. Essentially, improving our models' performance will require a blend of experimentation and deeper analysis of our data. This iterative refinement process is crucial for developing more accurate and practical predictive tools.

Bibliography

https://www.kaggle.com/datasets/bhavikjikadara/student-study-performance/data

Reference list

- *Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), 21-27.*
- *Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. Machine learning, 29(2-3), 103-130.*
- *Han, J., Kamber, M., & Pei, J. (2011). Data mining: Concepts and techniques (3rd ed.). Elsevier.*
- *Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.*
- *Mitchell, T. M. (1997). Machine learning. McGraw Hill.*
- Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to linear regression analysis*. John Wiley & Sons.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. Springer Science & Business Media.