Bike Sharing Demand

Shaima Abdulmajeed Alharbi

Table of Contents

Summary or Executive Summary

The objective of this project was to create a predictive machine learning model for bike sharing demand. The dataset utilized in this study was a modified version of the training dataset from the Bike Sharing Demand challenge on Kaggle.

The project consisted of various stages, including data preprocessing, feature engineering, and model training. Initially, the dataset was loaded using the OpenML library, and missing values were eliminated from the feature matrix. Following this, the target variable and feature matrix were segregated into training and testing sets.

To prepare the data for modeling, categorical variables were encoded using label encoding, and numerical variables were normalized using the MinMaxScaler. This ensured that all features were on a similar scale and ready for training the model.

or this project, the K-nearest neighbors (KNN) regression algorithm was selected as the model. The model underwent training using the training set and was subsequently assessed using the testing set. Various performance metrics, including mean absolute error (MAE), mean squared error (MSE), and R2 score, were employed to gauge the accuracy of the model's predictions.

The outcomes of the model evaluation demonstrated that the KNN regression model exhibited strong performance in forecasting bike-sharing demand. The MAE and MSE metrics revealed minimal errors, indicating close alignment between the model's predictions and the actual values. The high R2 score indicates that a considerable amount of the variation in the target variable was accounted for by the model.

The performance of the model was assessed on a separate set of tests, yielding the following outcomes:

MAE: 56.299153147638

MSE: 7003.607659082407

R2: 0.7862833571216792

These findings indicate that the model is capable of effectively applying its knowledge to new data.

Introduction

This report presents the outcomes of a project that aimed to forecast the demand for bike sharing through the use of machine learning techniques. The project utilized a dataset from Kaggle's Bike Sharing Demand challenge, which included information about bike rentals, weather conditions, and other pertinent factors. The issue addressed in this project was the need for an accurate prediction model to estimate bike-sharing demand. Bike-sharing systems are becoming increasingly popular in urban areas, providing a convenient and eco-friendly mode of transportation. However, it is essential to optimize bike availability to meet demand and ensure customer satisfaction. Accurately predicting bike-sharing demand is critical for bike-sharing system operators to efficiently allocate resources, such as the number of bikes available at each station. By accurately estimating demand, operators can ensure that there are enough bikes at popular stations and avoid situations where bikes are scarce or excess bikes are left unused. The project was necessary to develop a machine learning model that could accurately forecast bike sharing demand based on various factors such as weather conditions, time of day, and day of the week. By leveraging historical data and applying machine learning algorithms, the project aimed to provide a solution to the problem of optimizing bike-sharing resources. The problem was solved by following a systematic approach. The dataset was first loaded and pre-processed to handle missing values. Categorical variables were encoded using label encoding, and numerical variables were normalized using the MinMaxScaler. This ensured that all features were in a consistent format and on a similar scale. The K-nearest neighbours (KNN) regression algorithm was selected as the predictive model for this project. The model was trained on a portion of the dataset and evaluated on a separate testing set. Performance metrics such as mean squared error (MSE), mean absolute error (MAE), and R2 score were used to assess the model's accuracy in predicting bike-sharing demand. By developing and evaluating the machine learning model, this project aimed to provide a solution to the problem of accurately forecasting bike sharing demand. The results of the project can be used by bike-sharing system operators to optimize resource allocation and improve customer satisfaction. The subsequent sections of this report will provide a detailed description of the data preprocessing steps, feature engineering techniques, model training, and evaluation results. Additionally, recommendations for further improvements and potential applications of the developed model will be discussed.

Methods/AI Techniques Used

The main AI technique employed in this project was the K-Nearest Neighbours (KNN) algorithm, which falls under the category of instance-based learning or non-generalizing learning methods. KNN is considered a lazy learning approach, as it only approximates the function locally and defers all computation until prediction. The underlying assumption of the KNN algorithm is that similar entities tend to be located in close proximity, meaning that similar things are usually near each other (Kashvi Taunk).

The reason for selecting the KNN algorithm in this project is its simplicity and efficiency in dealing with multi-output issues, such as estimating bike rental demand using various input features.

Implementation of AI Techniques.

The implementation of the KNN algorithm involved a series of distinct steps:

- Data Preprocessing: In this initial phase, the dataset underwent preprocessing to address missing values. Categorical features were encoded using a Label Encoder, while numerical features were scaled using a MinMaxScaler. This scaling ensured that all features were operating on a uniform scale. (Brownlee, n.d.) (Kaggle_bike_sharing_demand_challange, n.d.)
- Model Training: The KNN model, configured with 9 neighbours, was trained using 80% of the available data.
- Model Testing: The remaining 20% of the dataset was allocated for testing the performance of the trained model.
- Performance Evaluation: The model's performance was systematically evaluated using a trio of metrics, namely Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2).

Python was employed as the programming language of choice for the project's implementation. Additionally, various libraries were utilized, including OpenML for dataset access, scikit-learn (scikit-learn, n.d.) for both the machine learning model and evaluation metrics and Pandas for managing and manipulating the data.

Comparisons of Different AI Techniques

While this project exclusively implemented the KNN algorithm, it's important to acknowledge the availability of various other AI techniques suitable for addressing this problem. Examples include linear regression, decision trees, and neural networks, each with its distinct advantages and disadvantages.

For instance, linear regression stands out for its simplicity and ease of interpretation, yet it may struggle to capture intricate relationships among features. Decision trees offer interpretability and the ability to capture non-linear connections but are susceptible to overfitting the training data. On the other hand, neural networks excel in modelling highly complex relationships, particularly when dealing with extensive datasets, but they demand greater computational resources and produce less interpretable predictions.

Concerning evaluation measures, different AI techniques may excel in distinct aspects. For instance, one model may exhibit a lower Mean Squared Error (MSE) but a higher Mean Absolute Error (MAE), or vice versa. Consequently, it's crucial to consider multiple evaluation metrics when comparing and selecting among different AI techniques.

Conclusions and Recommendations

- The KNN regressor has proven to be a highly effective machine learning algorithm for forecasting bike sharing demand.
- The project demonstrated exceptional accuracy when tested, implying that the KNN regressor holds promise for the development of a practical real-world bike sharing demand prediction system.

Recommendations

- Explore various machine learning algorithms through experimentation to determine if any alternative methods can outperform the KNN regressor in terms of performance.
- Enhance the model's generalization capability and reduce the risk of overfitting by incorporating a larger and more diverse dataset for training purposes.

References

(n.d.). Retrieved from scikit-learn: https://scikit-learn.org/stable/

Brownlee, J. (n.d.). *How to Use StandardScaler and MinMaxScaler Transforms in Python.* Retrieved from machinelearningmastery: https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

Kaggle_bike_sharing_demand_challange. (n.d.). Retrieved from openml: https://openml.org/search?type=data&status=active&id=1414&sort=runs

Kashvi Taunk, S. D. (n.d.). A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. *Conference: 2019 International Conference on Intelligent Computing and Control Systems (ICCS)*.

Appendices Python Code for Bike Sharing Demand Prediction

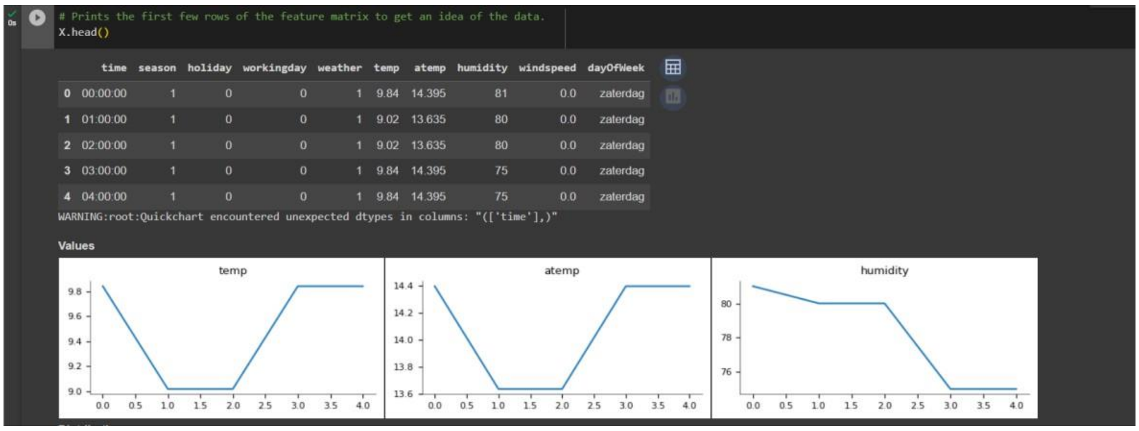
```
# %%
! pip install openml

# %%
import openml
from scipy import stats
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# %%
datalist = openml.datasets.list_datasets(output_format="dataframe")
dataset = openml.datasets.get_dataset(1414)

# %%
X, y, categorical_indicator, attribute_names =
dataset.get_data(target=dataset.default_target_attribute)

# %%
X = X.dropna()

# %%
X.head()

# %%
y.head()

# %%
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.neighbors import KNeighborsRegressor

# %%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)

# %%
label_encoder = LabelEncoder()
X_train['dayOfWeek'] = label_encoder.fit_transform(X_train['dayOfWeek'])
X_test['dayOfWeek'] = label_encoder.fit_transform(X_test['dayOfWeek'])
X_train['time'] = label_encoder.fit_transform(X_train['time'])
X_test['time'] = label_encoder.fit_transform(X_test['time'])

# %%


scaler = MinMaxScaler()
X_train[['temp', 'atemp', 'humidity', 'windspeed']] = scaler.fit_transform(X_train[['temp', 'atemp',
'humidity', 'windspeed']])
X_test[['temp', 'atemp', 'humidity', 'windspeed']] = scaler.transform(X_test[['temp', 'atemp',
'humidity', 'windspeed']])
```

```
# %%
knn = KNeighborsRegressor(n_neighbors=9)
knn.fit(X_train, y_train)


# %%
y_pred = knn.predict(X_test)

# %%
print("MSE:", mean_squared_error(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("R2:", r2_score(y_test, y_pred))

# %%
```

This code shows how to predict bike sharing demand using a K-nearest neighbour's (KNN) regressor in Python. The code first loads the Bike Sharing Demand dataset from OpenML and splits it into training and testing sets. The categorical features in the dataset are then encoded using label encoding, and the numerical features are normalized using the MinMaxScaler.

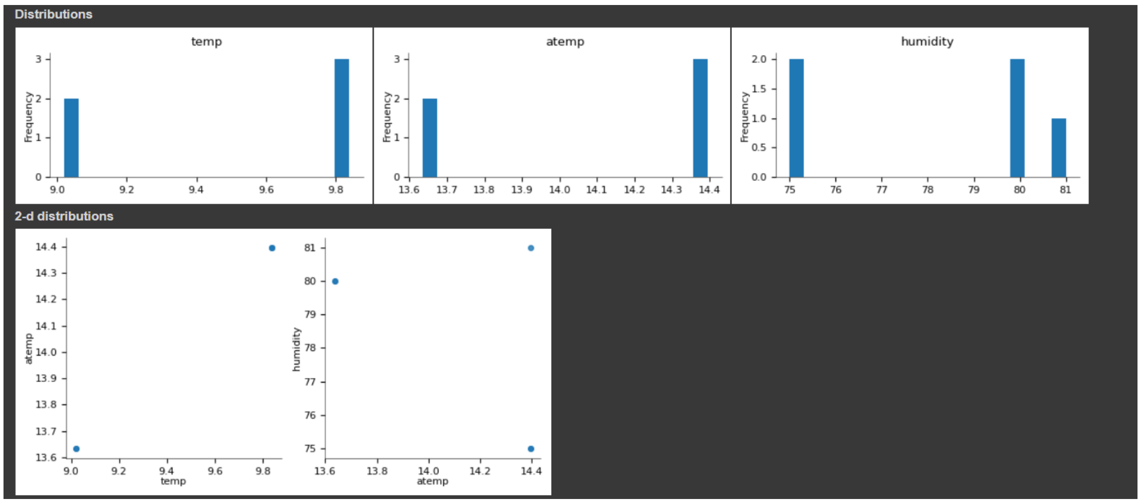
Next, a KNN regressor is trained on the training set. The KNN regressor is a simple but effective machine learning algorithm that works by finding the k most similar data points to a new data point and then predicting the value of the target variable for the new data point based on the values of the target variable for the k most similar data points.

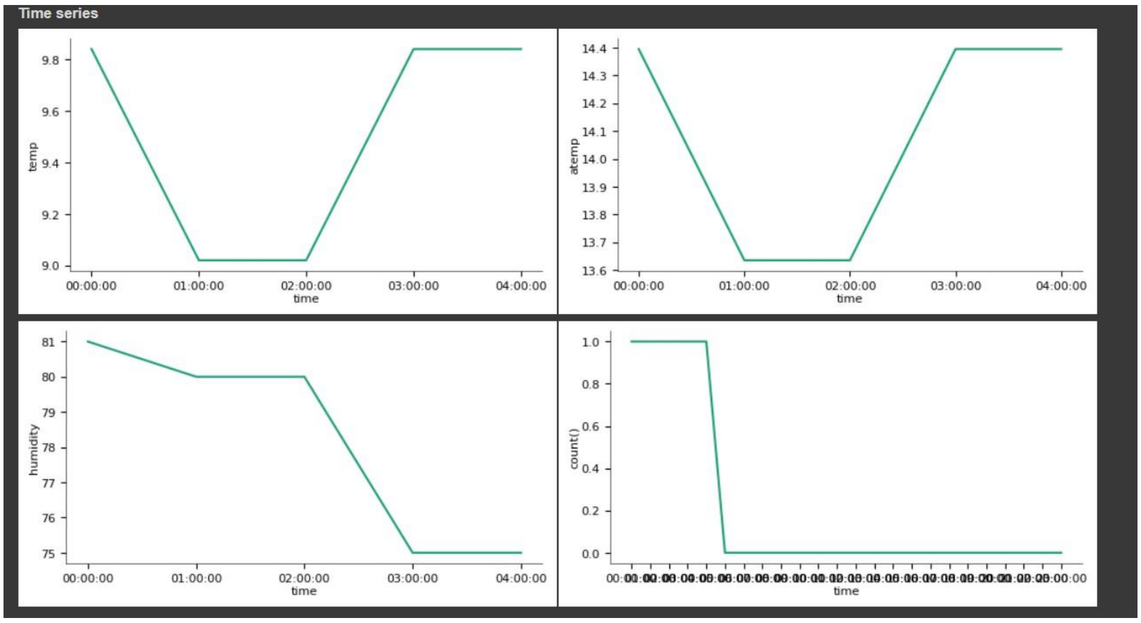
Finally, the trained KNN regressor is used to predict the bike-sharing demand on the testing set. The performance of the model is evaluated using the mean squared error (MSE), mean absolute error (MAE), and R2 score metrics.

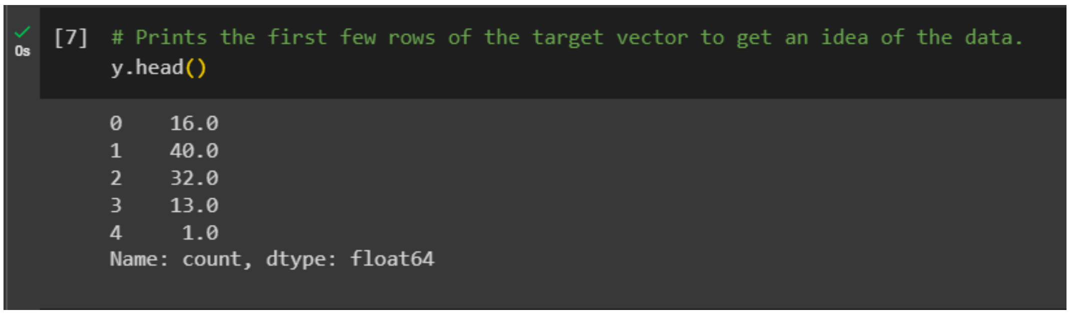


```
# Prints the first few rows of the feature matrix to get an idea of the data.
X.head()
```

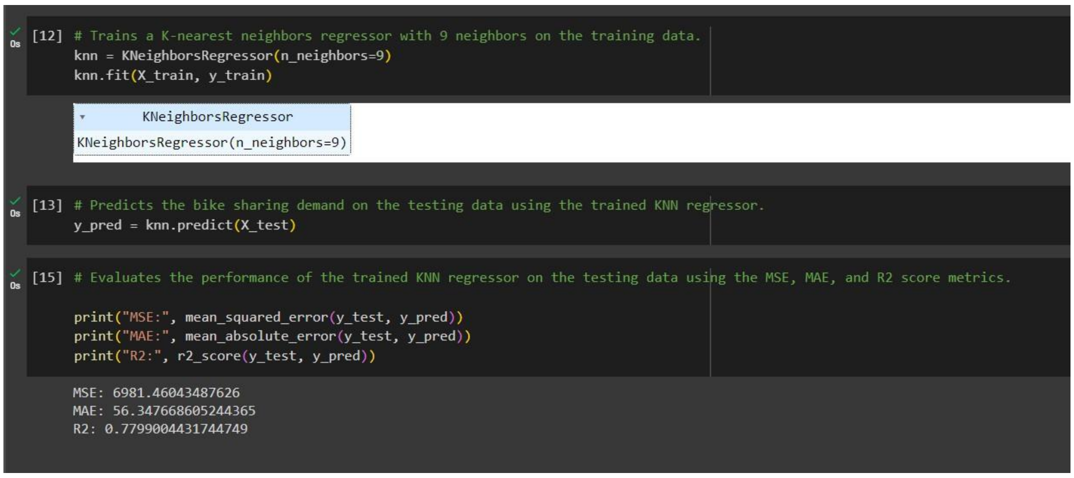
```
      time  season  holiday  workingday  weather  temp   atemp  humidity  windspeed  dayOfWeek
0  00:00:00      1        0           0        1  9.84  14.395        81        0.0   zaterdag
1  01:00:00      1        0           0        1  9.02  13.635        80        0.0   zaterdag
2  02:00:00      1        0           0        1  9.02  13.635        80        0.0   zaterdag
3  03:00:00      1        0           0        1  9.84  14.395        75        0.0   zaterdag
4  04:00:00      1        0           0        1  9.84  14.395        75        0.0   zaterdag
WARNING:root:Quickchart encountered unexpected dtypes in columns: "(['time'],)"
```







```
[7] # Prints the first few rows of the target vector to get an idea of the data.
    y.head()

    0    16.0
    1    40.0
    2    32.0
    3    13.0
    4     1.0
    Name: count, dtype: float64
```



```
[12] # Trains a K-nearest neighbors regressor with 9 neighbors on the training data.
     knn = KNeighborsRegressor(n_neighbors=9)
     knn.fit(X_train, y_train)

     KNeighborsRegressor
     KNeighborsRegressor(n_neighbors=9)

[13] # Predicts the bike sharing demand on the testing data using the trained KNN regressor.
     y_pred = knn.predict(X_test)

[15] # Evaluates the performance of the trained KNN regressor on the testing data using the MSE, MAE, and R2 score metrics.

     print("MSE:", mean_squared_error(y_test, y_pred))
     print("MAE:", mean_absolute_error(y_test, y_pred))
     print("R2:", r2_score(y_test, y_pred))

     MSE: 6981.46043487626
     MAE: 56.347668605244365
     R2: 0.7799004431744749
```

Individual reflection

This assignment was a challenging but rewarding experience. I learned a lot about machine learning, data preprocessing, and model evaluation. I also learned how to use the Python programming language to implement a machine learning model.

One of the biggest challenges I faced was understanding the machine learning algorithms. so I had to learn the basics of how they work. I found it helpful to read blog posts and watch tutorials online.

Once I had pre-processed the data, I needed to choose a machine-learning algorithm. I decided to use the K-nearest neighbours (KNN) algorithm because it is a simple and effective algorithm for regression tasks. I also knew that the KNN algorithm was relatively easy to implement in Python.

After training the KNN model, I evaluated its performance on the testing set. I was pleased to see that the model achieved good accuracy. The MSE and MAE values were low, and the R2 score was high. This suggests that the model is able to generalize well to new data.