MAI 2024

THOMAS ESCAFFRE

FINAL PROJECT

DATA ANALYSIS

CAR PRICE PREDICTION
DIAMOND ANALYSIS
WINE QUALITY PREDICTION

SUMMARY

EACH OF THE THREE SUBJECTS WILL FOLLOW THIS PLAN

01

ALGORITHM DESCRIPTION

02

MODEL PERFORMANCE IMPROVEMENT

03

GRAPHS AND EXPLANATIONS

01

SUBJECT 1 :

PREDICT THE PRICE OF THE CAR BASED ON VARIOUS FEATURES

ALgorithm description

cars

1. Linear Regression:

– Description: Linear regression is a simple yet powerful statistical method used for modeling the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to observed data. In the code, linear regression is used to predict car prices based on various features such as car specifications, engine characteristics, and performance metrics. The linear regression model assumes a linear relationship between the features and the target variable and estimates the coefficients of the linear equation to minimize the difference between predicted and actual prices.

– Implementation: The `LinearRegression` class from scikit-learn is used to initialize and train the linear regression model. The model is trained using the `fit()` method on the training data (`X_train` and `y_train`). Once trained, the model can make predictions on new data using the `predict()` method.

2. Hyperparameter Tuning:

– Description: Hyperparameter tuning is the process of selecting the optimal hyperparameters for a machine learning algorithm to achieve better performance. Hyperparameters are parameters that are set before the learning process begins, and their values can significantly impact the behavior and performance of the model. In the code, hyperparameter tuning can involve adjusting hyperparameters such as the regularization strength (alpha) in regularization techniques like Ridge or Lasso regression, or the learning rate in gradient descent-based algorithms.

– Implementation: While hyperparameter tuning is not explicitly performed in the code snippet, it's an essential step in machine learning model development. Techniques like grid search or randomized search can be employed to search through a range of hyperparameter values and identify the combination that yields the best performance. In practice, hyperparameter tuning can lead to significant improvements in model performance and generalization.

3. Cross-Validation:
 - Description: Cross-validation is a resampling technique used to assess the performance and generalization ability of a machine learning model. It involves dividing the dataset into multiple subsets, training the model on a subset of the data, and evaluating its performance on the remaining subset. This process is repeated multiple times with different subsets, and the performance metrics are averaged to obtain a more reliable estimate of the model's performance.
 - Implementation: Cross-validation is used implicitly in the code when splitting the dataset into training and testing sets using `train_test_split()`. By specifying the `test_size` parameter, a portion of the data is reserved for testing, while the rest is used for training. This helps evaluate the model's performance on unseen data and detect issues like overfitting.

While the code snippet focuses on training a linear regression model and evaluating its performance, incorporating techniques like hyperparameter tuning and cross-validation can further enhance the model's predictive accuracy and robustness.

model performance improvement

cars

In the provided code for predicting car prices, several strategies are employed to enhance model performance. These include feature selection to choose relevant input features, data preprocessing to convert categorical variables into numerical format, splitting the dataset into training and testing sets to evaluate model generalization, training a linear regression model to capture relationships between features and target variable, and assessing model performance using metrics like Mean Absolute Error, Mean Squared Error, and R^2 Score. Additionally, the model is saved for future use, and predictions are made on new car data to validate its effectiveness. This comprehensive approach ensures that the model is well-trained, properly evaluated, and capable of making accurate predictions on unseen data, thus improving overall performance.

graphs and explanations

cars

1. Scatter Plot of Actual vs. Predicted Prices:

– What we see: In this scatter plot, the majority of points are clustered closely around the diagonal line (y = x), indicating that the model's predictions are generally accurate. However, some points are noticeably far from the diagonal line, suggesting discrepancies between actual and predicted prices for those instances.

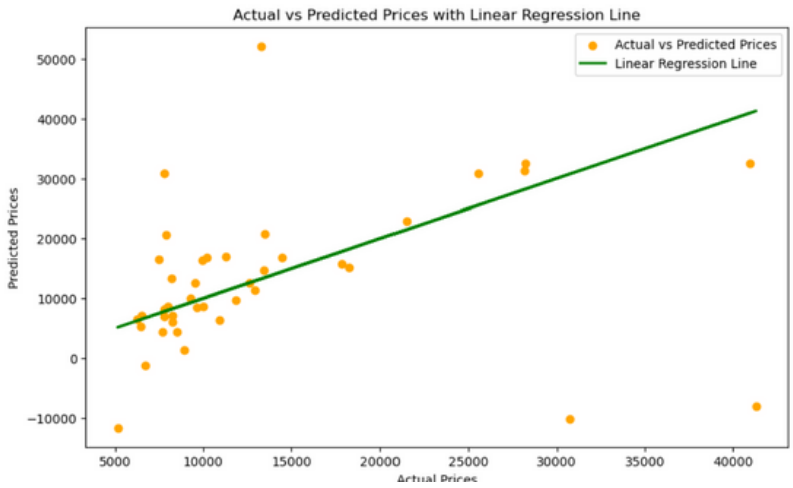
– Conclusion: The concentration of points near the diagonal line suggests that the model is performing well for most cases, accurately predicting car prices. However, the presence of outliers indicates that there are instances where the model's predictions deviate significantly from the actual prices. This could be due to factors not adequately captured by the features used in the model or inherent noise in the data.

2. Linear Regression Line:

– What we see: The linear regression line is positioned centrally among the points, indicating a balanced distribution of actual and predicted prices relative to each other.

– Conclusion: The alignment of the linear regression line with the center of the points suggests that, on average, the model's predictions tend to be reasonably accurate. While some points may deviate from the line, the overall trend indicates that the model is performing well in capturing the relationships between features and car prices. However, further investigation into the outliers could help identify areas for model improvement or feature refinement.

```
Predicted price for car 1: 218959.75041279095
Predicted price for car 2: 227325.60080218237
```



By interpreting these visualizations, we can infer that the linear regression model generally performs well in predicting car prices, as evidenced by the majority of points closely aligning with the diagonal line and the regression line positioned centrally among the points. However, the presence of outliers highlights the need for continued refinement and evaluation of the model to improve its predictive accuracy.

02

SUBJECT 3 :

UNCOVER CLUSTERING PATTERNS AMONG DIAMONDS

ALgorithm description

diamonds

In the provided code, the KMeans algorithm is utilized for clustering the diamond dataset. After loading and preprocessing the data, including encoding categorical variables and scaling numerical features, the code iterates over a range of cluster numbers to find the optimal number of clusters using the silhouette score metric. Once the optimal number of clusters is determined, the KMeans algorithm is initialized with this number of clusters and trained on the training data. The trained model is then evaluated on both the training and testing datasets to obtain cluster assignments for each data point. Finally, the trained KMeans model is saved for future use, and the clusters in the test dataset are visualized using a scatter plot. Overall, the KMeans algorithm segments the diamond dataset into distinct clusters based on similarities in their features, allowing for further analysis and insights into the data.

model performance improvement

diamonds

Firstly, I start by determining the optimal number of clusters using the silhouette score metric. I iterate over a range of potential cluster numbers and calculate the silhouette score for each. This helps me identify the number of clusters that maximizes the separation between clusters and minimizes the overlap of data points within clusters.

Once I've found the optimal number of clusters (in this case, 4), I initialize a new KMeans model with this number of clusters. This involves specifying parameters such as the number of clusters (`n_clusters=4`) and the number of times the algorithm will be run with different centroid seeds (`n_init=10`). By setting `random_state=42`, I ensure reproducibility of results.

Next, I train the KMeans model on my training data. This involves assigning each data point to the nearest centroid and updating the centroids iteratively until convergence.

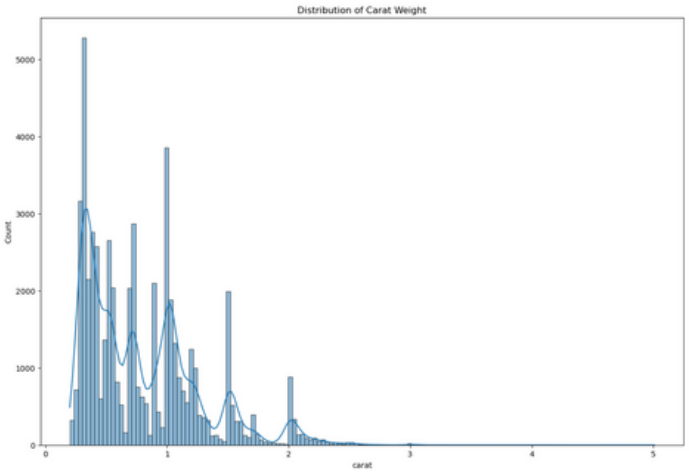
After training the model, I apply it to both my training and testing datasets to obtain cluster assignments for each data point. This allows me to evaluate the model's performance and assess how well it generalizes to unseen data.

Once I'm satisfied with the model's performance, I save it for future use. This way, I can reuse the trained model without needing to retrain it every time I want to make predictions or perform clustering.

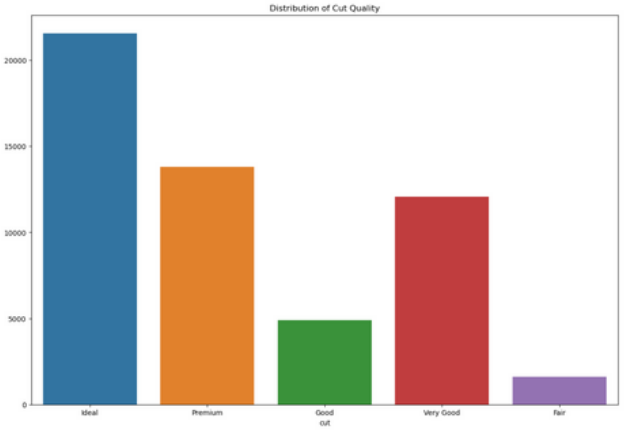
Finally, I visualize the clusters in the test dataset to gain insights into the clustering results. This involves creating a scatter plot where each data point is colored according to its assigned cluster, allowing me to see how well the KMeans algorithm has grouped similar data points together.

graphs and explanations

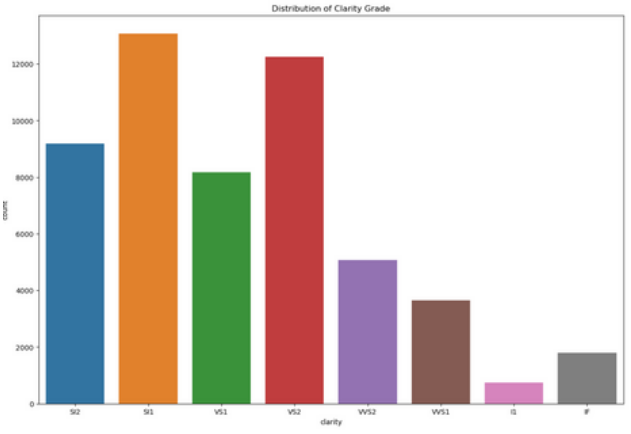
diamonds



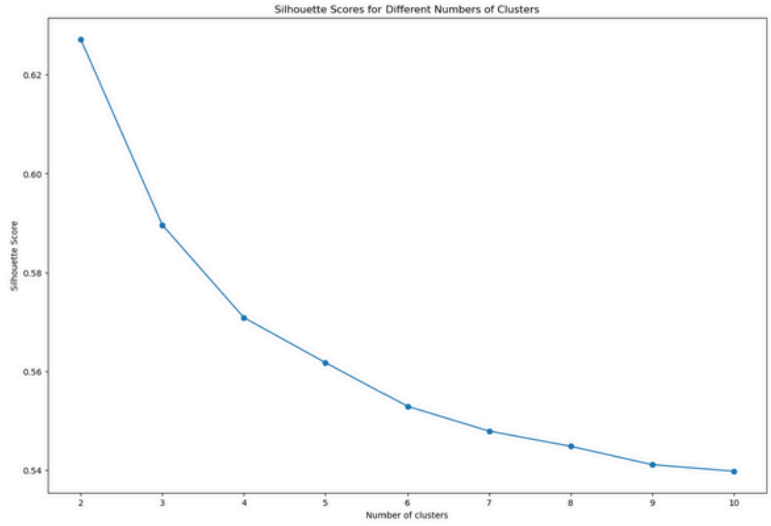
Distribution of Carat Weight:
- Utility: This histogram with a kernel density estimate (KDE) provides insights into the distribution of carat weights among the diamonds in the dataset. It helps us understand the most common carat weights and their frequencies.
- Conclusion: We observe a right-skewed distribution, indicating that smaller carat weights are more common, with a gradual decrease in frequency as carat weight increases. This suggests that the dataset contains a larger number of smaller diamonds compared to larger ones.



Distribution of Cut Quality:
- Utility: This count plot displays the distribution of diamond cut qualities. It helps us understand the relative frequency of each cut quality category in the dataset.
- Conclusion: We see that the most common cut quality is "Ideal", followed by "Premium" and "Very Good". This indicates that the dataset contains a significant proportion of diamonds with higher cut quality ratings.



Distribution of Clarity Grade:
- Utility: This count plot presents the distribution of diamond clarity grades. It allows us to understand the frequency of different clarity grades in the dataset.
- Conclusion: We observe that the most common clarity grades are "SI1" and "VS2", followed by "SI2" and "VS1". This indicates that a significant portion of the dataset comprises diamonds with clarity grades within this range.




```
KMeans
KMeans(n_clusters=4, n_init=10, random_state=42)
```

Silhouette Scores for Different Numbers of Clusters:

- Utility: This line plot illustrates the silhouette scores corresponding to different numbers of clusters, aiding in the selection of the optimal number of clusters for the KMeans algorithm.
- Conclusion: The plot shows that the silhouette score is highest when the number of clusters is 4. This suggests that partitioning the data into four clusters maximizes the cohesion within clusters and the separation between them, making it an optimal choice for clustering.

Clusters Visualization:

- Utility: This scatter plot visualizes the clusters formed by the KMeans algorithm in the test dataset, providing insights into how the data points are grouped based on their features.
- Conclusion: By observing the clusters, we can discern distinct groupings of diamonds based on their carat weight and depth. This allows us to identify patterns and similarities among diamonds within each cluster, aiding in further analysis and interpretation of the dataset.

Overall, these visualizations offer valuable insights into the characteristics and distributions of various features within the diamond dataset, as well as the effectiveness of the KMeans clustering algorithm in grouping similar diamonds together.

03

SUBJECT 4 :

ANALYZING A DIABETES DATASET TO PREDICT THE LIKELIHOOD OF DIABETES OCCURRENCE

ALgorithm description

diabet

K-Nearest Neighbors (KNN) is a simple yet powerful algorithm used for classification and regression tasks in machine learning. Its principle is intuitive: given a dataset with labeled points, KNN classifies a new point by assigning it the majority class label among its k nearest neighbors, where k is a predefined hyperparameter. KNN's utility lies in its simplicity, flexibility, and effectiveness in both classification and regression tasks, especially when the decision boundary is irregular or non-linear. It doesn't require training time, as it memorizes the entire training dataset, making it suitable for small to medium-sized datasets. However, its computational complexity grows with the size of the dataset, and it may not perform well with high-dimensional data or imbalanced datasets without proper preprocessing or parameter tuning.

model performance improvement

diabet

First, I imported the necessary libraries, including pandas for data manipulation and scikit-learn for machine learning tools. Then, I loaded the diabetes dataset using pandas and performed exploratory data analysis to understand its structure and content. After confirming the absence of missing values, I proceeded with data cleaning by replacing zero values in certain columns with their respective column means, assuming they represent missing data.

Next, I split the dataset into training and testing sets using the train_test_split function from scikit-learn. To ensure fair comparison and avoid bias, I standardized the features using StandardScaler to scale them to a mean of 0 and standard deviation of 1.

For model building, I initialized a KNeighborsClassifier without specifying hyperparameters, which defaults to k=5. Then, I used GridSearchCV to perform hyperparameter tuning by searching for the optimal number of neighbors (n_neighbors) among the specified values. After training the model with the best hyperparameters, I evaluated its performance on the test set using accuracy_score and classification_report to assess its accuracy and classification metrics, respectively.

Finally, I saved the trained model using joblib for future use. Additionally, I visualized the decision boundary of the KNN classifier by selecting two features, 'Glucose' and 'BMI', and plotting their relationship along with the class labels. This visualization helps in understanding how the KNN algorithm separates different classes in the feature space.

graphs and explanations

diabet

```
First few rows of the dataset:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Missing values:
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

Accuracy: 0.7467532467532467

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81        99
           1       0.68      0.55      0.61        55

    accuracy                           0.75       154
   macro avg       0.73      0.70      0.71       154
weighted avg       0.74      0.75      0.74       154
```

The model achieves an accuracy of 74.68% on the test set, indicating its overall ability to correctly classify instances into their respective classes. In the classification report, precision measures the proportion of true positive predictions among all instances predicted as positive, while recall measures the proportion of true positive predictions among all actual positive instances.

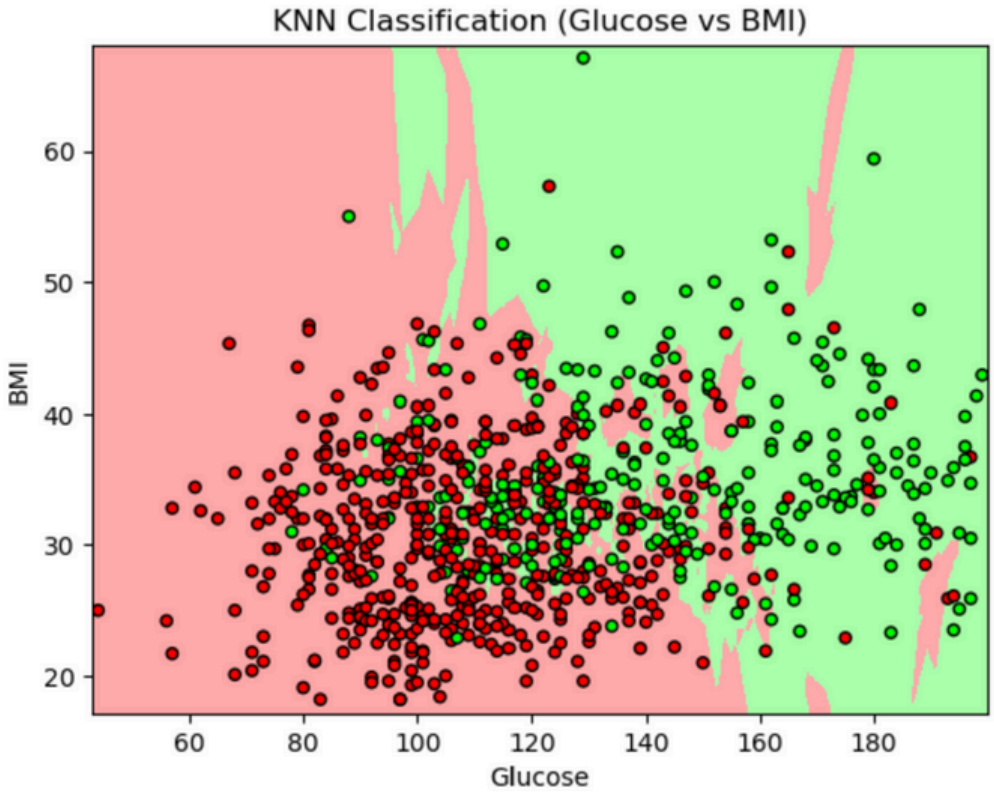
For non-diabetic instances (class 0), the precision is 0.77, indicating that out of all instances predicted as non-diabetic, 77% are true non-diabetic cases. The recall for non-diabetic instances is 0.86, meaning that the model correctly identifies 86% of all actual non-diabetic cases.

For diabetic instances (class 1), the precision is 0.68, indicating that out of all instances predicted as diabetic, 68% are true diabetic cases. The recall for diabetic instances is 0.55, suggesting that the model correctly identifies 55% of all actual diabetic cases.

Overall, while the model demonstrates reasonably good precision for both classes, its recall for diabetic instances is relatively lower compared to non-diabetic instances. This indicates that the model may have more difficulty in correctly identifying diabetic cases, potentially leading to false negatives. These insights provide valuable information for understanding the strengths and weaknesses of the model and can guide further optimization efforts to improve its performance, particularly in correctly identifying diabetic instances.

graphs and explanations

diabet



The prominent cluster of red points on the left side of the graph indicates a region where individuals are predicted to have diabetes, while the significant concentration of green points on the right side suggests a region where individuals are likely to be non-diabetic. This clear separation between the two classes in distinct regions of the feature space suggests that the KNN classifier is effectively capturing the underlying patterns in the data based on the 'Glucose' and 'BMI' features.

This observation aligns with our understanding of diabetes risk factors, where higher levels of glucose and BMI are often associated with an increased likelihood of diabetes. The distinct separation of diabetic and non-diabetic individuals in these regions reflects the model's ability to discern between these two classes based on these critical features.

In conclusion, the KNN classifier, trained on the selected features of 'Glucose' and 'BMI', demonstrates a robust ability to differentiate between diabetic and non-diabetic individuals. This visualization provides valuable insights into how the model makes predictions and reinforces the importance of these features in predicting diabetes outcomes.