Assignment-based Subjective Questions

1. From your analysis of the categorical variables from the dataset, what could you infer about their effect on the dependent variable? (3 marks)

In the Boom Bikes dataset, the categorical variables typically include:

Season: This variable indicates the season in which the bike rentals occurred (e.g., Spring, Summer, Fall, Winter).

Year: This variable indicates the year of the rental (e.g., 2018, 2019).

Month: This variable indicates the month of the rental (e.g., January, February, etc.).

Holiday: This variable indicates whether the day is a holiday or not.

Weekday: This variable indicates the day of the week (e.g., Monday, Tuesday, etc.).

Working Day: This variable indicates whether the day is a working day or not.

Weather Situation: This variable describes the weather conditions (e.g., Clear, Mist, Light Rain/Snow).

Inferences about Their Effect on the Dependent Variable (Bike Demand)

Season: Bike rentals tend to vary significantly with the season. For example, rentals are generally higher in the Fall and Summer due to favorable weather conditions, while Winter sees a drop in demand.

Year: The dataset might show an increase in bike rentals from one year to the next, indicating growing popularity or increased availability of bikes.

Month: Certain months, particularly those in the Fall and Summer, may see higher bike rentals. For instance, July and September might have higher median rentals compared to other months.

Holiday: Bike rentals are typically higher on holidays as people have more leisure time to rent bikes for recreational purposes.

Weekday: The day of the week can influence bike rentals. For example, rentals might be higher on weekends (Thursday, Friday, and Sunday) when people are more likely to engage in leisure activities.

Working Day: There might be a slight increase in bike rentals on working days as people use bikes for commuting purposes.

Weather Situation: Weather conditions have a significant impact on bike rentals. Clear weather is most optimal for bike rentals, while adverse weather conditions like rain or snow can reduce demand.

These inferences help in understanding the patterns and factors influencing bike demand, allowing for better business strategies and operational adjustments.

2. Why is it important to use drop_first=True during dummy variable creation? (2 mark)

Using drop_first=True when creating dummy variables is important to avoid the dummy variable trap. The dummy variable trap occurs when the dummy variables are highly correlated (multicollinear), which can lead to issues in regression models. Here's why it's important:

Multicollinearity: When you create dummy variables for a categorical feature with (n) categories, you get (n) dummy variables. However, these (n) variables are not independent; they are perfectly correlated. This multicollinearity can cause problems in regression models, making it difficult to estimate the coefficients accurately.

Redundancy: Including all (n) dummy variables introduces redundancy. One of the dummy variables can be perfectly predicted from the others, which doesn't add any new information to the model.

Interpretability: Dropping the first dummy variable (or any one of them) helps in interpreting the model coefficients. The dropped category serves as the baseline, and the coefficients of the remaining dummy variables represent the change relative to this baseline.

For example, if you have a categorical variable "Color" with three categories: Red, Blue, and Green, creating dummy variables without drop_first=True would result in three dummy variables: Color_Red, Color_Blue, and Color_Green. This can lead to multicollinearity. By setting drop_first=True, you drop one dummy variable (e.g., Color_Red), and the model will use it as the baseline.

Here's how you can create dummy variables with drop_first=True in Python:

Python

```
import pandas as pd

# Sample data
data = pd.DataFrame({
    'Color': ['Red', 'Blue', 'Green', 'Red', 'Blue']
})

# Create dummy variables with drop_first=True
dummies = pd.get_dummies(data['Color'], drop_first=True)

print(dummies)
```

AI-generated code. Review and use carefully. More info on FAQ.

This will result in:

	Blue	Green
0	0	0
1	1	0
2	0	1
3	0	0
4	1	0

In this case, Color_Red is dropped, and the model will use it as the baseline category.

3. Looking at the pair-plot among the numerical variables, which one has the highest correlation with the target variable? (1 mark)

To determine which numerical variable has the highest correlation with the target variable, you can look at the pair plot and calculate the correlation coefficients. Here's how you can do it in Python using Seaborn and Pandas:

Create a Pair Plot: This will help visualize the relationships between the numerical variables and the target variable.

Calculate Correlation Coefficients: This will quantify the strength of the relationships.

Here's a sample code to achieve this:

Python

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('bike_sharing_data.csv')

# Create a pair plot
sns.pairplot(data, vars=['temperature', 'humidity', 'windspeed', 'demand'])
plt.show()
```

```
# Calculate correlation coefficients

correlation_matrix = data[['temperature', 'humidity', 'windspeed', 'demand']].corr()

print(correlation_matrix)


# Identify the variable with the highest correlation with the target variable

highest_correlation = correlation_matrix['demand'].drop('demand').idxmax()

print(f'The variable with the highest correlation with demand is: {highest_correlation}')
```

AI-generated code. Review and use carefully. More info on FAQ.

This code will:

Visualize the relationships between the numerical variables and the target variable using a pair plot.

Calculate the correlation coefficients to identify which variable has the highest correlation with the target variable (demand).

The output will show the correlation matrix, and the variable with the highest correlation with demand will be printed.

4. How did you validate the assumptions of Linear Regression after building the model on the training set? (3 marks)

Validating the assumptions of linear regression is crucial to ensure the reliability and accuracy of the model. Here are the key assumptions and how you can validate them:

Linearity:

Validation: Check if the relationship between the independent variables and the dependent variable is linear.

How: Plot the residuals versus the predicted values. If the plot shows a random scatter, the linearity assumption is likely met.

Python

```
import matplotlib.pyplot as plt

plt.scatter(y_pred, y_test - y_pred)

plt.xlabel('Predicted Values')

plt.ylabel('Residuals')

plt.title('Residuals vs Predicted Values')

plt.show()
```

AI-generated code. Review and use carefully. More info on FAQ.

Independence:

Validation: Ensure that the residuals are independent.

How: Use the Durbin-Watson test to check for autocorrelation in the residuals.

Python

```
from statsmodels.stats.stattools import durbin_watson

dw_stat = durbin_watson(y_test - y_pred)

print(f'Durbin-Watson statistic: {dw_stat}')
```

AI-generated code. Review and use carefully. More info on FAQ.

Homoscedasticity:

Validation: Check if the residuals have constant variance.

How: Plot the residuals versus the predicted values. If the spread of residuals is constant across all levels of predicted values, the assumption is met.

Python

```
plt.scatter(y_pred, y_test - y_pred)

plt.xlabel('Predicted Values')

plt.ylabel('Residuals')

plt.title('Residuals vs Predicted Values')

plt.show()
```

AI-generated code. Review and use carefully. More info on FAQ.

Normality:

Validation: Ensure that the residuals are normally distributed.

How: Use a Q-Q plot to check the normality of residuals. Additionally, you can perform the Shapiro-Wilk test.

Python

```
import scipy.stats as stats

import matplotlib.pyplot as plt

stats.probplot(y_test - y_pred, dist="norm", plot=plt)

plt.show()
```

```
from scipy.stats import shapiro

shapiro_test = shapiro(y_test - y_pred)

print(f'Shapiro-Wilk test statistic: {shapiro_test[0]}, p-value: {shapiro_test[1]}')
```

AI-generated code. Review and use carefully. More info on FAQ.

By validating these assumptions, you can ensure that your linear regression model is robust and reliable. If any of these assumptions are violated, you may need to consider alternative modeling techniques or transform your data to meet the assumptions.

5. Based on the final model, which are the top 3 features contributing significantly towards explaining the demand of the shared bikes? (2 marks)

To identify the three features that contribute significantly to explaining the demand for shared bikes, you can look at the coefficients of the linear regression model. Features with larger absolute values of coefficients have a greater impact on the target variable.

Here's how you can extract and interpret the coefficients from the linear regression model:

Python

```
# Assuming 'model' is your trained LinearRegression model

coefficients = pd.DataFrame({

    'Feature': X_train.columns,

    'Coefficient': model.coef_

})


# Sort the coefficients by their absolute values in descending order

coefficients['Absolute Coefficient'] = coefficients['Coefficient'].abs()

coefficients = coefficients.sort_values(by='Absolute Coefficient', ascending=False)


# Display the top 3 features

top_3_features = coefficients.head(3)

print(top_3_features)
```

AI-generated code. Review and use carefully. More info on FAQ.

This code will output the top three features with the highest absolute coefficients, indicating their significant contribution to the demand for shared bikes.

General Subjective Questions

1. Explain the linear regression algorithm in detail. (4 marks)

Linear regression is a fundamental algorithm in statistics and machine learning used to model the relationship between a dependent variable (target) and one or more independent variables (features). Here's a detailed explanation:

1. Concept

Linear regression aims to find the best-fitting linear relationship between the dependent variable (y) and the independent variables (X). The relationship is modeled using a linear equation:

[y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon]

where:

(y) is the dependent variable.

(x_1, x_2, \ldots, x_n) are the independent variables.

(\beta_0) is the intercept.

(\beta_1, \beta_2, \ldots, \beta_n) are the coefficients (slopes) for the independent variables.

(\epsilon) is the error term (residual).

2. Assumptions

Linear regression relies on several key assumptions:

Linearity: The relationship between the dependent and independent variables is linear.

Independence: Observations are independent of each other.

Homoscedasticity: The residuals (errors) have constant variance at every level of the independent variables.

Normality: The residuals are normally distributed.

3. Steps to Perform Linear Regression

a. Data Collection and Preparation

Collect and preprocess the data, ensuring it is clean and suitable for modeling. This includes handling missing values, outliers, and creating dummy variables for categorical features.

b. Splitting the Data

Divide the data into training and testing sets to evaluate the model's performance on unseen data.

c. Fitting the Model

Use the training data to fit the linear regression model. The goal is to find the coefficients ((\beta)) that minimize the sum of squared residuals (errors).

d. Making Predictions

Use the fitted model to make predictions on the test data.

e. Evaluating the Model

Evaluate the model's performance using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared ((R^2)).

4. Mathematical Approach

a. Ordinary Least Squares (OLS)

The most common method to estimate the coefficients is Ordinary Least Squares (OLS). OLS minimizes the sum of the squared differences between the observed and predicted values:

[\text{Minimize} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2]

where (y_i) is the actual value and (\hat{y}_i) is the predicted value.

b. Gradient Descent

Another approach is Gradient Descent, an iterative optimization algorithm that adjusts the coefficients to minimize the cost function (sum of squared residuals). The update rule for the coefficients is:

[\beta_j := \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}]

where (\alpha) is the learning rate and (J(\beta)) is the cost function.

5. Interpreting the Results

Coefficients: Indicate the change in the dependent variable for a one-unit change in the independent variable, holding other variables constant.

Intercept: Represents the expected value of the dependent variable when all independent variables are zero.

R-squared: Measures the proportion of variance in the dependent variable explained by the independent variables.

6. Example in Python

Here's a simple example of linear regression using Python:

Python

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Load the dataset
data = pd.read_csv('bike_sharing_data.csv')

# Define independent variables (features) and dependent variable (target)
X = data[['temperature', 'humidity', 'windspeed']]
y = data['demand']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Training
model = LinearRegression()
model.fit(X_train, y_train)

# Model Predictions
```

```
y_pred = model.predict(X_test)


# Model Evaluation
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)


print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'R-squared: {r2}')
```

AI-generated code. Review and use carefully. More info on FAQ.

This code demonstrates how to build, train, and evaluate a linear regression model using Python.

2. Explain the Anscombe's quartet in detail. (3 marks)

Anscombe's quartet

Explore

Anscombe's quartet is a set of four datasets that were created by the statistician Francis Anscombe in 1973 to demonstrate the importance of graphing data before analyzing it. Despite having nearly identical simple descriptive statistics, these datasets appear very different when graphed. This highlights how relying solely on statistical measures can be misleading without visualizing the data.

Key Points of Anscombe's Quartet

Identical Statistical Properties:

All four datasets have the same mean, variance, correlation, and linear regression line.

For example, they all have the same mean of (x) and (y), the same variance of (x) and (y), the same correlation coefficient, and the same linear regression equation.

Different Distributions:

When plotted, each dataset reveals a different pattern, showing that they are not as similar as their statistics suggest.

The differences include linear relationships, non-linear relationships, and the presence of outliers.

The Four Datasets

Dataset I:

Appears to have a simple linear relationship between (x) and (y).

Fits a linear regression model well.

Dataset II:

Shows a clear non-linear relationship.

A linear regression model is not appropriate for this dataset.

Dataset III:

Contains an outlier that significantly affects the linear regression line.

The outlier skews the results, demonstrating the impact of influential points.

Dataset IV:

Has a high-leverage point that creates a high correlation coefficient.

The other data points do not indicate any relationship, but the single high-leverage point distorts the analysis.

Importance of Anscombe's Quartet

Visualization: Emphasizes the need to visualize data before performing statistical analysis. Graphs can reveal patterns, relationships, and anomalies that statistics alone might miss.

Outliers and Influential Points: Highlights how outliers and influential points can affect statistical measures and regression models.

Model Appropriateness: Demonstrates that different datasets may require different types of models, and a one-size-fits-all approach can be misleading.

Example Visualization

Here's how you can visualize Anscombe's quartet using Python:

Python

```
import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd


# Load Anscombe's quartet dataset

anscombe = sns.load_dataset("anscombe")


# Create a grid of plots
```

```
sns.set(style="ticks")

g = sns.FacetGrid(anscombe, col="dataset", col_wrap=2)

g.map(sns.scatterplot, "x", "y")

g.map(sns.lineplot, "x", "y", ci=None)


# Show the plots

plt.show()
```

AI-generated code. Review and use carefully. More info on FAQ.

This code will generate scatter plots for each of the four datasets, allowing you to see the differences visually.

Anscombe's quartet serves as a powerful reminder of the importance of data visualization and the potential pitfalls of relying solely on statistical summaries. It underscores the need for a comprehensive approach to data analysis that includes both numerical and graphical methods123.

3. What is Pearson's R? (3 marks)

Pearson correlation coefficient

Explore

Pearson's (r), also known as the Pearson correlation coefficient, is a measure of the linear relationship between two quantitative variables. It quantifies the strength and direction of this relationship, providing a value between -1 and 1:

1 indicates a perfect positive linear relationship.

-1 indicates a perfect negative linear relationship.

0 indicates no linear relationship.

Formula

The formula for Pearson's (r) is:

$$ r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}} $$

where:

(n) is the number of data points.

(x) and (y) are the two variables being compared.

(\sum) denotes the summation.

Interpretation

Positive Correlation: As one variable increases, the other variable also increases.

Negative Correlation: As one variable increases, the other variable decreases.

No Correlation: There is no linear relationship between the variables.

Example

If you have data on students' study hours and their exam scores, Pearson's (r) can help determine if there is a linear relationship between the amount of time spent studying and the exam scores.

Visualization

Visualizing the data with a scatter plot can help understand the relationship. The closer the data points are to a straight line, the stronger the linear relationship.

When to Use

Pearson's (r) is appropriate when:

Both variables are continuous.

The relationship between the variables is linear.

The data is normally distributed.

Example in Python

Here's how you can calculate Pearson's (r) using Python:

Python

```
import pandas as pd

import numpy as np

from scipy.stats import pearsonr

# Sample data

data = {'study_hours': [1, 2, 3, 4, 5], 'exam_scores': [50, 55, 60, 65, 70]}

df = pd.DataFrame(data)
```

```
# Calculate Pearson's r
corr, _ = pearsonr(df['study_hours'], df['exam_scores'])
print(f'Pearson correlation coefficient: {corr}')
```

AI-generated code. Review and use carefully. More info on FAQ.

This code will output the Pearson correlation coefficient, indicating the strength and direction of the linear relationship between study hours and exam scores.

4. What is scaling? Why is scaling performed? What is the difference between normalized scaling and standardized scaling? (3 marks)

Scaling is a data preprocessing technique used in machine learning to transform feature values to a similar scale. This ensures that all features contribute equally to the model, which is particularly important for algorithms sensitive to the scale of data, such as those using gradient descent or distance metrics (e.g., KNN, K-means).

Why is Scaling Performed?

Scaling is performed to:

Improve Model Performance: Ensures that features with larger values do not dominate the learning process.

Enhance Convergence: Helps gradient descent algorithms converge faster by ensuring uniform step sizes.

Reduce Bias: Prevents bias towards features with larger magnitudes.

Handle Outliers: Reduces the impact of outliers on the model.

Normalized Scaling vs. Standardized Scaling

Both normalization and standardization are types of scaling, but they differ in their approach:

Normalized Scaling

Definition: Scales data to a specific range, typically between 0 and 1.

Formula: $X_{norm} = \frac{X_{max} - X_{min}}{X - X_{min}}$

Use Case: Useful when you want to ensure that all features are on the same scale, especially when there are no significant outliers.

Example: If you have a feature with values ranging from 10 to 100, normalization will scale these values to a range of 0 to 1.

Standardized Scaling

Definition: Scales data to have a mean of 0 and a standard deviation of 1.

Formula: $Xstd=\sigma X-\mu$

Use Case: Useful when the data follows a Gaussian distribution or when you want to maintain the distribution of the data.

Example: If you have a feature with a mean of 50 and a standard deviation of 10, standardization will transform the data to have a mean of 0 and a standard deviation of 1.

Key Differences

Range: Normalization scales data to a fixed range (0 to 1), while standardization scales data based on the mean and standard deviation.

Outliers: Normalization is sensitive to outliers, whereas standardization is less affected by them.

Distribution: Normalization does not assume any specific distribution of data, while standardization assumes a Gaussian distribution12.

5. You might have observed that sometimes the value of VIF is infinite. Why does this happen? (3 marks)

The Variance Inflation Factor (VIF) is a measure used to detect multicollinearity in a regression model. It quantifies how much the variance of a regression coefficient is inflated due to multicollinearity with other predictors.

Why VIF Can Be Infinite

An infinite VIF value indicates perfect multicollinearity, meaning one predictor variable is a perfect linear combination of one or more other predictor variables. This situation arises when there is exact collinearity among the predictors, leading to the following issues:

Singular Matrix: The design matrix (X) becomes singular, meaning it cannot be inverted. This is a mathematical requirement for calculating regression coefficients.

Redundant Information: Perfect multicollinearity implies that one predictor does not provide any new information beyond what is already provided by other predictors.

Example Scenario

Consider a dataset with two predictors, (X_1) and (X_2), where (X_2 = 2 \times X_1). In this case, (X_2) is perfectly collinear with (X_1), leading to an infinite VIF for both predictors.

How to Address Infinite VIF

Remove Redundant Predictors: Identify and remove one of the perfectly collinear predictors.

Combine Predictors: Combine collinear predictors into a single predictor through techniques like Principal Component Analysis (PCA).

Regularization: Use regularization techniques like Ridge Regression, which can handle multicollinearity by adding a penalty term to the regression.

6. What is a Q-Q plot? Explain the use and importance of a Q-Q plot in linear regression. (3 marks)

A Q-Q plot (Quantile-Quantile plot) is a graphical tool used to compare the distribution of a dataset to a theoretical distribution, such as the normal distribution. It helps to visually assess whether the data follows a specified distribution.

How a Q-Q Plot Works

Quantiles Calculation: The quantiles of the observed data are plotted against the quantiles of the theoretical distribution.

Line of Best Fit: If the data follows the theoretical distribution, the points will approximately lie on a straight line (usually the 45-degree line).

Use and Importance in Linear Regression

In the context of linear regression, a Q-Q plot is primarily used to check the normality of residuals. Here's why it's important:

Assumption Check: Linear regression assumes that the residuals (errors) are normally distributed. A Q-Q plot helps verify this assumption.

Model Diagnostics: By examining the Q-Q plot, you can identify deviations from normality, such as skewness or kurtosis, which may indicate issues with the model.

Improving Model Fit: If the residuals are not normally distributed, transformations or different modeling techniques may be needed to improve the model fit.

Interpreting a Q-Q Plot

Straight Line: If the points lie on or near the straight line, the residuals are approximately normally distributed.

Deviations: Systematic deviations from the line suggest departures from normality. For example:

S-shaped curve: Indicates heavy tails (leptokurtic distribution).

Inverted S-shaped curve: Indicates light tails (platykurtic distribution).

Curvature: Indicates skewness in the data.

Example

Imagine you have a dataset of residuals from a linear regression model. You create a Q-Q plot to compare these residuals to a normal distribution. If the points closely follow the 45-degree line, it suggests that the residuals are normally distributed, validating one of the key assumptions of linear regression123.