**Assignment – 2**

1)What is the primary objective of data wrangling?

a) Data visualization

b) Data cleaning and transformation

c) Statistical analysis

d) Machine learning modelling

The primary objective of data wrangling is data cleaning and transformation. This process involves converting and mapping raw data into another format that's more useful, accessible, and easier to analyze. It's an essential step in data science projects and is often required before data analysis and modeling can take place.

import pandas as pd

# create a sample dataframe with missing values and inconsistent data

data = {

'Name': ['Jai', 'Princi', 'Gaurav', 'Anuj', 'Anuj', 'Dhiraj', 'Hitesh'],

'Age': [27, 24, 22, np.nan, 32, 12, 52],

'Address': ['Nagpur', 'Kanpur', 'Allahabad', 'Kannuaj', 'Kannuaj', 'Allahabad', 'Kannuaj'],

'Qualification': ['Msc', 'MA', 'MCA', 'Phd', 'Phd', 'Bcom', 'B.hons'],

'Mobile No': [97.0, 91.0, 58.0, np.nan, 2000.0, 3000.0, 4000.0],

'Salary': [np.nan, np.nan, 1000.0, 2000.0, 2000.0, 3000.0, 4000.0]

}

df = pd.DataFrame(data)

# replace missing values with column mean

df['Age'] = df['Age'].fillna(df['Age'].mean())

# replace missing values in Mobile No column with median

df['Mobile No'] = df['Mobile No'].fillna(df['Mobile No'].median())

# encode categorical variables

df['Qualification'] = df['Qualification'].astype('category').cat.codes

# filter out rows where Salary is less than 3000

df = df[df['Salary'] >= 3000]

# display the resulting dataframe

print(df)

2. Explain the technique used to convert categorical data into numerical data. How does it help in data analysis?

To convert categorical data into numerical data, you can use various techniques in Python, such as using the cat.codes attribute in Pandas, LabelEncoder class from the sklearn.preprocessing module, or the replace() method in Pandas.

Here's an example using LabelEncoder:

python

Edit

Full Screen

Copy code

from sklearn.preprocessing import LabelEncoder

import pandas as pd

# Create a DataFrame with categorical data

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

# Convert categorical data to numerical data using LabelEncoder

le = LabelEncoder()

df['Color'] = le.fit\_transform(df['Color'])

# View the converted DataFrame

print(df)

Output:

Edit

Full Screen

Copy code

Color

0 2

1 0

2 1

3 2

4 1

Converting categorical data to numerical data is helpful in data analysis because many machine learning algorithms require numeric inputs. By converting categorical data to numeric values, you can use these algorithms to analyze the data and make predictions.

Moreover, using numerical representations of categorical data can also help in data visualization, such as when creating scatter plots or line graphs. Numerical data is also easier to analyze using statistical methods, such as calculating means, medians, and standard deviations.

Another technique to convert categorical data to numerical data is one-hot encoding, which can be done using pd.get\_dummies() in Pandas. Here's an example:

python

Edit

Full Screen

Copy code

import pandas as pd

# generate df with 1 col and 4 rows

data = {

"fruit": ["apple", "banana", "orange", "apple"]

}

# one-hot-encode using pandas

df = pd.DataFrame(data)

df\_encoded = pd.get\_dummies(df["fruit"])

# View the encoded DataFrame

print(df\_encoded)

Output:

Edit

Full Screen

Copy code

apple banana orange

0 1 0 0

1 0 1 0

2 0 0 1

3 1 0 0

One-hot encoding creates binary columns for each category, indicating the presence or absence of a category for each observation. This technique can be useful when the categorical data has a large number of unique categories, as it avoids introducing an arbitrary ordering between categories.

3. How does LabelEncoding differ from OneHotEncoding?

LabelEncoding and OneHotEncoding are both techniques used to convert categorical data into numerical data, but they differ in how they assign numerical values to the categories.

LabelEncoding assigns a unique integer value to each category based on alphabetical or numerical order. This can be problematic when the categories have an inherent order or hierarchy, as it may introduce bias or incorrect relationships between the categories. For example, if a categorical variable has categories "Low", "Medium", and "High", assigning integer values 1, 2, and 3 respectively would imply that the difference between "Low" and "Medium" is the same as the difference between "Medium" and "High", which may not be true.

On the other hand, OneHotEncoding creates binary variables for each category, where each category is represented as a binary vector with a 1 in the position corresponding to that category and 0 in all other positions. This creates a sparse matrix with high dimensionality, but it avoids introducing any bias or incorrect relationships between the categories.

In summary, LabelEncoding is a simpler and more space-efficient technique, but it can introduce bias or incorrect relationships between categories. OneHotEncoding is more robust and unbiased, but it can create a sparse matrix with high dimensionality. The choice between the two techniques depends on the specific use case and the trade-offs between simplicity, space efficiency, and robustness.

4. Describe a commonly used method for detecting outliers in a dataset. Why is it important to identify outliers?

a commonly used method for detecting outliers in a dataset. One such method is the Z-score method, which is a statistically based approach for outlier detection. It computes the standard score, or Z-score, for each data point, which measures how many standard deviations a data point deviates from the mean of the dataset. We then set a threshold for our Z-score, and data points with Z-scores greater than it are considered outliers.

Here's some example Python code that uses the Z-score method to detect outliers in a dataset:

python

Edit

Full Screen

Copy code

from scipy import stats

# Assume `data` is a pandas DataFrame containing the dataset

threshold = 2.5

z\_scores = stats.zscore(data)

outliers = data[abs(z\_scores) > threshold]

Identifying outliers is important for several reasons. Firstly, outliers can significantly impact the results of statistical analyses and machine learning algorithms, leading to inaccurate conclusions or predictions. By identifying and removing outliers, we can improve the accuracy and reliability of our analyses.

Secondly, outliers can provide valuable insights into the data, such as identifying errors or anomalies that may require further investigation. For example, in a dataset of customer orders, an outlier might represent a large order that requires special attention or a data entry error that needs to be corrected.

Finally, outliers can affect the visualization of the data, making it difficult to identify patterns and trends. By detecting and removing outliers, we can create more accurate and informative visualizations that better represent the underlying data.

In summary, the Z-score method is a commonly used method for detecting outliers in a dataset, and identifying outliers is important for ensuring the accuracy and reliability of statistical analyses and machine learning algorithms, providing valuable insights into the data, and creating more accurate and informative visualizations.

5. Explain how outliers are handled using the Quantile Method.

The Quantile Method is another commonly used method for detecting and handling outliers in a dataset. Unlike the Z-score method, which is based on the mean and standard deviation of the dataset, the Quantile Method is based on the distribution of the data.

The basic idea behind the Quantile Method is to divide the dataset into equally sized groups, called quantiles, based on their rank order. For example, if we divide the dataset into 4 equal-sized groups, we get the quartiles, with the first quartile (Q1) representing the 25th percentile of the data, the second quartile (Q2) representing the 50th percentile (or median), and the third quartile (Q3) representing the 75th percentile.

Once we have calculated the quartiles, we can use them to define a range for each group. Specifically, we can define an outlier as any data point that falls outside of the range [Q1 - 1.5IQR, Q3 + 1.5IQR], where IQR is the interquartile range, defined as the difference between the third and first quartiles (Q3 - Q1).

Here's some example Python code that uses the Quantile Method to detect outliers in a dataset:

python

Edit

Full Screen

Copy code

import numpy as np

# Assume `data` is a pandas DataFrame containing the dataset

Q1 = np.percentile(data, 25)

Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5\*IQR

upper\_bound = Q3 + 1.5\*IQR

outliers = data[(data < lower\_bound) | (data > upper\_bound)]

Once we have identified the outliers, we can handle them in several ways, depending on the specific use case and the reason for their presence. Some common ways to handle outliers include:

Removing them from the dataset: This is the simplest and most common way to handle outliers, but it should be done with caution, as it can lead to loss of information and biased results.

Imputing them with a value: We can replace the outliers with a value that is more representative of the dataset, such as the median or mean. This can be useful when the outliers are due to measurement errors or missing data.

Transforming the data: We can apply a non-linear transformation, such as a logarithmic or square root transformation, to reduce the impact of the outliers and make the data more normally distributed.

Using robust statistical methods: We can use statistical methods that are less sensitive to outliers, such as the median absolute deviation (MAD) or the trimmed mean, instead of the mean and standard deviation.

In summary, the Quantile Method is a commonly used method for detecting and handling outliers in a dataset, based on the distribution of the data. Outliers can be handled in several ways, depending on the specific use case and the reason for their presence, including removing them, imputing them with a value, transforming the data, or using robust statistical methods.

6. Discuss the significance of a Box Plot in data analysis. How does it aid in identifying potential outliers?

A box plot, also known as a box-and-whisker plot, is a standardized way of displaying the distribution of a dataset based on five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can provide a lot of statistical information, including the median, ranges, and outliers, in a single plot.

In a box plot, the box extends from Q1 to Q3, and the line in the middle of the box represents the median. The "whiskers" represent the range of the data, with the lower whisker extending to the minimum and the upper whisker extending to the maximum, unless there are outliers. Outliers are defined as any data point that falls below Q1 - 1.5 \* IQR or above Q3 + 1.5 \* IQR, where IQR is the interquartile range (Q3 - Q1). These outliers are typically displayed as individual points outside the whiskers.

Here's an example of how to create a box plot with outliers in Python using the seaborn library:

python

Edit

Full Screen

Copy code

import seaborn as sns

import matplotlib.pyplot as plt

# create example data

data = [0, 1, 2, 3, 6, 6, 6, 10]

# create box plot with outliers

sns.boxplot(data=data)

plt.show()

And here's an example of how to create a box plot with no outliers:

python

Edit

Full Screen

Copy code

import seaborn as sns

import matplotlib.pyplot as plt

# create example data with no outliers

data = [0, 1, 2, 3, 4, 5, 6]

# create box plot with no outliers

sns.boxplot(data=data)

plt.show()

Box plots can help identify potential outliers in a dataset, which can be important to investigate further as they may represent errors or unusual observations that require further attention. By visualizing the distribution of the data and looking for outliers, we can gain insights into the data and identify any patterns or anomalies that may not be immediately apparent from looking at the raw data alone.

Here's an example of how to identify potential outliers in Python using the numpy library:

python

Edit

Full Screen

Copy code

import numpy as np

# create example data

data = [0, 1, 2, 3, 6, 6, 6, 10]

# calculate five number summary

q1, q3 = np.percentile(data, [25, 75])

iqr = q3 - q1

lower\_bound = q1 - 1.5 \* iqr

upper\_bound = q3 + 1.5 \* iqr

# identify potential outliers

outliers = [x for x in data if x < lower\_bound or x > upper\_bound]

print("Outliers:", outliers)

This will output:

yaml

Edit

Full Screen

Copy code

Outliers: [10]

In summary, box plots are a powerful tool for visualizing the distribution of a dataset and identifying potential outliers. By understanding how box plots work and how to interpret them, we can gain valuable insights into our data and make more informed decisions.

7. What type of regression is employed when predicting a continuous target variable?

When predicting a continuous target variable, linear regression is typically employed. Linear regression is a type of regression analysis that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. The dependent variable in linear regression is continuous, meaning it can take on any value within a range, and the independent variables can be either continuous or categorical. The goal of linear regression is to find the best-fitting line that describes the relationship between the independent and dependent variables, allowing for predictions to be made about the dependent variable based on the values of the independent variables.

8. Identify and explain the two main types of regression.

The two main types of regression are linear regression and logistic regression.

Linear regression is used when the dependent variable is continuous and the relationship between the dependent and independent variables is linear. It involves finding the best-fitting line through the data points and uses a linear equation to make predictions. However, linear regression is susceptible to outliers and should not be used with large datasets that have a high correlation between independent variables.

Logistic regression, on the other hand, is used when the dependent variable has a discrete value, meaning it can only have one of two values. Logistic regression uses a sigmoid curve to show the relationship between the target and independent variables, and works best with large datasets that have an almost equal occurrence of values in target variables. However, it can create a problem when ranking the variables if there is a high correlation between independent variables.

It is important to choose the right type of regression analysis, as it can unlock the full potential of the data and set you on the path to greater insights.

9. When would you use Simple Linear Regression? Provide an example scenario.

Simple linear regression is used when you want to estimate the relationship between two continuous variables. It attempts to fit a straight line that describes the relationship between the independent variable (x) and the dependent variable (y) in the form of an equation, y = mx + b.

For example, let's say you work for a car company and you want to determine if there is a relationship between the engine size (in liters) and the fuel efficiency (measured in miles per gallon) of cars. You can collect data on a sample of cars, and then use simple linear regression to determine if there is a significant relationship between engine size and fuel efficiency. The equation produced by the regression analysis could then be used to predict fuel efficiency for cars with different engine sizes.

It's important to note that simple linear regression assumes that the relationship between the variables is linear, and that the errors are normally distributed and independent. If these assumptions are not met, other types of regression analysis may be more appropriate.

10. In Multi Linear Regression, how many independent variables are typically involved?

n multiple linear regression, there are typically two or more independent variables involved. These independent variables, along with the dependent variable, are all quantitative and are used to estimate the relationship between them. The formula for multiple linear regression includes multiple independent variables, each with its own regression coefficient that represents the estimated effect of that variable on the dependent variable.

11. When should Polynomial Regression be utilized? Provide a scenario where Polynomial Regression would be preferable over Simple Linear Regression

Polynomial regression is used when the relationship between the dependent variable and the independent variable(s) is not linear, but instead curvilinear. In polynomial regression, one or more of the independent variables are raised to a power greater than one, allowing for a more flexible model that can fit complex, non-linear relationships.

For example, let's say you are a researcher studying the relationship between the dosage of a medication (in milligrams) and the resulting blood pressure (in mmHg) of patients. You collect data on a sample of patients and plot the data, but you notice that the relationship between the dosage and blood pressure is not linear. Instead, the blood pressure decreases with increasing dosage up to a certain point, and then starts to increase again. In this case, polynomial regression would be preferable over simple linear regression because it can capture the non-linear relationship between the dosage and blood pressure.

Specifically, you might choose to fit a quadratic regression model to the data, which would include a squared term for the dosage variable. This would allow the model to fit a curve that captures the initial decrease and subsequent increase in blood pressure with increasing dosage. Other types of polynomial regression, such as cubic or quartic regression, can be used to model even more complex relationships.

12. What does a higher degree polynomial represent in Polynomial Regression? How does it affect the model's complexity?

In polynomial regression, a higher degree polynomial represents a more complex relationship between the dependent variable and the independent variable(s). The degree of the polynomial is the highest power of the independent variable(s) in the regression equation. For example, a degree 2 polynomial would have terms up to x^2, while a degree 3 polynomial would have terms up to x^3.

The degree of the polynomial affects the model's complexity by increasing the number of parameters that need to be estimated. A higher degree polynomial can fit more complex relationships, but it also has a greater risk of overfitting the data, especially if there are too few data points or if the degree is chosen arbitrarily.

Here's an example of how to fit a polynomial regression model with a higher degree using Python and scikit-learn library:

python

Edit

Full Screen

Copy code

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

# Generate some random data

x = np.linspace(0, 10, 100)

y = np.sin(x) + np.random.normal(0, 1, 100)

# Create a polynomial feature matrix with degree 3

poly = PolynomialFeatures(degree=3)

x\_poly = poly.fit\_transform(x.reshape(-1, 1))

# Fit a polynomial regression model with degree 3

lr = LinearRegression()

lr.fit(x\_poly, y)

# Plot the data and the regression line

plt.scatter(x, y)

plt.plot(x, lr.predict(poly.transform(x.reshape(-1, 1))), label='Degree 3')

plt.legend()

plt.show()

In this example, we generate some random data y as a sine function with added noise, and fit a polynomial regression model of degree 3 to it. We use the PolynomialFeatures class from scikit-learn to create a polynomial feature matrix of degree 3, which we then use to fit the regression model. The resulting model can capture the complex relationship between x and y with a high degree of accuracy.

13. Highlight the key difference between Multi Linear Regression and Polynomial Regression

The key difference between multiple linear regression and polynomial regression is the functional form of the relationship between the dependent variable and the independent variables.

Multiple linear regression assumes a linear relationship between the dependent variable and each of the independent variables. The regression equation takes the form:

y = b0 + b1 \* x1 + b2 \* x2 + ... + bp \* xp

where y is the dependent variable, x1, x2, ..., xp are the independent variables, and b0, b1, b2, ..., bp are the regression coefficients.

On the other hand, polynomial regression allows for a non-linear relationship between the dependent variable and the independent variables. The regression equation takes the form:

y = b0 + b1 \* x + b2 \* x^2 + ... + bd \* x^d

where x is the independent variable, y is the dependent variable, and b0, b1, b2, ..., bd are the regression coefficients. The degree of the polynomial (d) determines the complexity of the relationship between x and y.

In other words, multiple linear regression assumes that the relationship between the variables is a straight line, while polynomial regression allows for a curved relationship. Polynomial regression can be seen as a generalization of multiple linear regression, where the independent variables can be transformed to higher powers to fit more complex relationships.

Here's an example to illustrate the difference between multiple linear regression and polynomial regression using Python and scikit-learn library:

python

Edit

Full Screen

Copy code

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

# Generate some random data

x = np.linspace(0, 10, 100)

y = 2 \* x + np.random.normal(0, 1, 100)

# Fit a multiple linear regression model

lr = LinearRegression()

lr.fit(x.reshape(-1, 1), y)

# Fit a polynomial regression model of degree 2

poly = PolynomialFeatures(degree=2)

x\_poly = poly.fit\_transform(x.reshape(-1, 1))

lr\_poly = LinearRegression()

lr\_poly.fit(x\_poly, y)

# Plot the data and the regression lines

plt.scatter(x, y)

plt.plot(x, lr.predict(x.reshape(-1, 1)), label='Multiple Linear Regression')

plt.plot(x, lr\_poly.predict(poly.transform(x.reshape(-1, 1))), label='Polynomial Regression (Degree 2)')

plt.legend()

plt.show()

In this example, we generate some random data y as a linear function with added noise, and fit both a multiple linear regression model and a polynomial regression model of degree 2 to it. We use the PolynomialFeatures class from scikit-learn to create a polynomial feature matrix of degree 2, which we then use to fit the polynomial regression model. The resulting polynomial regression model can capture the non-linear relationship between x and y with a higher degree of accuracy than the multiple linear regression model.

14. Explain the scenario in which Multi Linear Regression is the most appropriate regression technique

Multiple linear regression is used when you want to model the relationship between two or more independent variables and one dependent variable. It is the most appropriate regression technique when you want to investigate the effect of multiple variables on a single outcome variable, and when the relationship between the variables is linear.

For example, let's say you are a researcher studying the factors that influence students' academic performance in college. You have data on several variables, including the number of hours per week a student spends studying, the number of hours per week a student spends working a part-time job, and the student's high school GPA. You want to know how these variables are related to the student's college GPA.

In this scenario, multiple linear regression is the most appropriate regression technique because you have two or more independent variables (hours spent studying, hours spent working, and high school GPA) and one dependent variable (college GPA), and you want to investigate the linear relationship between these variables. Multiple linear regression will allow you to estimate the effect of each independent variable on the dependent variable, while controlling for the effects of the other independent variables. This will help you understand which factors are most strongly associated with academic performance in college, and whether these relationships are linear or not.

It's important to note that multiple linear regression assumes that the relationship between the independent variables and the dependent variable is linear, and that the errors are normally distributed and independent. If these assumptions are not met, other types of regression analysis may be more appropriate.

15. What is the primary goal of regression analysis?

The primary goal of regression analysis is to estimate the relationships between a dependent variable and one or more independent variables. It can be used to assess the strength of the relationship between variables and to model the future relationship between them. Regression analysis can help us understand which factors are most strongly associated with the outcome variable, and whether these relationships are linear or not. By estimating these relationships, we can make predictions about the outcome variable based on the values of the independent variables. Regression analysis is widely used in many fields, including finance, to study the relationships between variables and to make predictions based on those relationships.