**Interview Questions:**

1.What is Normalization & Standardization and how is it helpful?

**Normalization and Standardization:**

**Normalization**

* **Definition**:  
  Scaling all the data values of a feature to fit within a specific range, typically **[0, 1]**.
* **Example**:  
  Let us take as comparing two features:
  + Feature A: Age (ranges from 20 to 60).
  + Feature B: Income (ranges from 20,000 to 100,000).

Without normalization, the large range of income could dominate the smaller range of age, making the machine learning model pay more attention to income.

* **Working and Formula with example**:

x′=x−minmax−minx' = \frac{x - \text{min}}{\text{max} - \text{min}}x′=max−minx−min​

Let us say:

* + Age: 30, Min: 20, Max: 60.
  + Normalized Age = 30−2060−20=0.25\frac{30 - 20}{60 - 20} = 0.2560−2030−20​=0.25.

Here, all features are scaled between 0 and 1, making them equally important for the model.

**Standardization**

* **Definition**:  
  Rescaling the data so that it has a mean of **0** and a standard deviation of **1**.
* **Example**:  
  If one feature has very high values (e.g., income) and another has small values (e.g., age), the model may struggle to learn effectively. Standardization ensures all features are on the same scale but doesn’t squash them into a fixed range.
* **Working and Formula with example**:

z=x−μσz = \frac{x - \mu}{\sigma}z=σx−μ​

Example:

* + Age: 30, Mean: 40, Standard Deviation: 10.
  + Standardized Age = 30−4010=−1.0\frac{30 - 40}{10} = -1.01030−40​=−1.0.

Here, age is expressed as how many standard deviations it is from the mean.

**Another Example**

* Suppose we are training a machine learning model to predict whether someone buys a product based on their **age** and **income**:
  + Age ranges from 18 to 80.
  + Income ranges from $20,000 to $200,000.

If we don’t normalize or standardize:

* The model might treat income as more important because its values are much larger.

If we normalize:

* Age becomes something like 0.1 to 1.0, and income becomes 0.1 to 1.0. Now they’re comparable!

If we standardize:

* Age becomes values like -2 to 2 (standardized around 0), and income becomes values like -2 to 2. Again, they’re on the same scale.

**Main Differences**

* **Normalization**: Use when we want values between a fixed range (e.g., 0 to 1).
* **Standardization**: Use when we want data centered at 0 with a standard deviation of 1.

**Example:**

* **Normalization** is like resizing all images in a photo album to fit a 4x6 frame, regardless of their original dimensions.
* **Standardization** is like adjusting all photos to have the same brightness and contrast for consistency.

Bottom of Form

2.What techniques can be used to address multicollinearity in multiple linear regression?

**Techniques to Address Multicollinearity in Multiple Linear Regression**

Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated. This can make it challenging to estimate the coefficients accurately and interpret the model. Below are some common techniques to address multicollinearity, explained with Python scripts :

**1. Check for Multicollinearity**

Before addressing multicollinearity, it is essential to detect it. The Variance Inflation Factor (VIF) is a common diagnostic tool.

import pandas as pd

import numpy as np

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

# Create a function to calculate VIF

def calculate\_vif(dataframe):

vif\_data = pd.DataFrame()

vif\_data["Feature"] = dataframe.columns

vif\_data["VIF"] = [variance\_inflation\_factor(dataframe.values, i) for i in range(dataframe.shape[1])]

return vif\_data

# Example dataset

data = pd.DataFrame({

'X1': np.random.rand(100),

'X2': np.random.rand(100) \* 2,

'X3': lambda x: x['X1'] \* 0.8 + x['X2'] \* 0.2 # Simulate multicollinearity

})

# Calculate VIF

vif = calculate\_vif(data)

print(vif)

**2. Remove Highly Correlated Predictors**

If two variables are highly correlated, consider removing one of them.

# Calculate the correlation matrix

correlation\_matrix = data.corr()

# Display correlation matrix

print(correlation\_matrix)

# Drop one of the highly correlated variables (e.g., X3)

data\_reduced = data.drop(columns=['X3'])

**3. Principal Component Analysis (PCA)**

PCA transforms the predictors into a set of orthogonal components, eliminating multicollinearity while retaining most of the variance.

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

# Standardize the features

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data)

# Apply PCA

pca = PCA(n\_components=2) # Reduce to 2 components

pca\_data = pca.fit\_transform(scaled\_data)

print(f"Explained Variance Ratio: {pca.explained\_variance\_ratio\_}")

**4. Ridge Regression**

Ridge regression adds a penalty term to the loss function, which helps reduce the impact of multicollinearity.

from sklearn.linear\_model import Ridge

from sklearn.model\_selection import train\_test\_split

# Split data into features and target

X = data.drop(columns=['X1']) # Example: 'X1' as the target

y = data['X1']

# Split 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)

# Fit Ridge Regression

ridge = Ridge(alpha=1.0)

ridge.fit(X\_train, y\_train)

print(f"Ridge Coefficients: {ridge.coef\_}")

**5. Feature Engineering**

* Combine correlated variables into a single feature.
* Create interaction terms or polynomial features to represent the relationships more compactly.

# Combine X1 and X2 into a new feature

data['Combined'] = data['X1'] + data['X2']

data\_reduced = data.drop(columns=['X1', 'X2'])

**Assumptions and Implications**

1. **Assumption**: Linear relationships exist between predictors and the response variable.
   * Implication: Nonlinear relationships can lead to poor model performance. Consider transformation if needed.
2. **Assumption**: No perfect multicollinearity.
   * Implication: Perfectly correlated variables must be addressed (e.g., by dropping one variable).
3. **Assumption**: Normality and homoscedasticity of residuals.
   * Implication: If violated, consider robust regression techniques.
4. **Assumption**: Feature selection aligns with the model objective.
   * Implication: Dropping or transforming variables might reduce interpretability.

Bottom of Form