MACHINE LEARNING

In Q1 to Q11, only one option is correct, choose the correct option:

1. Which of the following methods do we use to find the best fit line for data in Linear Regression?

Ans: A) Least Square Error

2. Which of the following statements is true about outliers in linear regression?

Ans: A) Linear regression is sensitive to outliers.

3. A line falls from left to right if a slope is?

Ans: B) Negative Slope

4. Which of the following will have symmetric relation between dependent variable and independent variable?

Ans: B) Correlation

5. Which of the following is the reason for over-fitting conditions?

Ans: C) Low bias and high variance .

6. If output involves label then that model is called as:

Ans: B) Predictive modal

7. Lasso and Ridge regression techniques belong to___________?

Ans: D) Regularization

8. To overcome with imbalance dataset which technique can be used?

Ans: D) SMOTE

9. The AUC Receiver Operator Characteristic (AUCROC) curve is an evaluation metric for binary classification problems. It uses_______to make graph?

Ans: A) TPR and FPR

10. In AUC Receiver Operator Characteristic (AUCROC) curve for the better model area under the curve should be less.

Ans: B) False

11. Pick the feature extraction from below:

Ans: B) Apply PCA to project high dimensional data

In Q12, more than one options are correct, choose all the correct options:

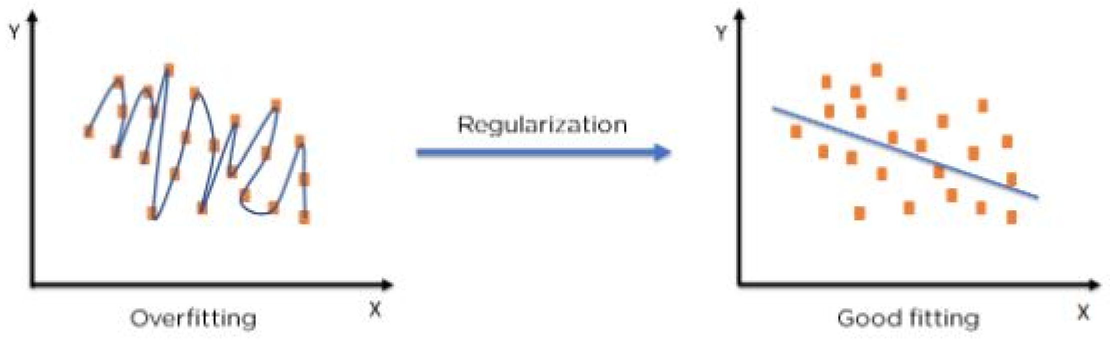
12. Which of the following is true about Normal Equation used to compute the coefficient of the Linear Regression?

Ans: A) We don't have to choose the learning rate.
B) It becomes slow when number of features is very large.

Q13 and Q15 are subjective answer type questions, Answer them briefly.

13. Explain the term regularization?

Ans: Regularization is a technique used in machine learning to prevent overfitting and improve the generalization performance of a model. Overfitting occurs when a model learns the noise in the training data rather than the underlying pattern or relationship between the input and output variables, resulting in poor performance.



Regularization helps to address this problem by adding a penalty term to the loss function of the model during training.. The penalty term discourages the model from assigning too much weight to any single feature or parameter, which can help to reduce the complexity of the model and prevent overfitting.

Regularization works by adding a penalty or complexity term to the complex model. Let's consider the simple linear regression equation:
y= β0+β1x1+β2x2+β3x3+⋯+ β nxn +b
In the above equation, Y represents the value to be predicted
X1, X2, …Xn are the features for Y.
β0,β1,…..βn are the weights or magnitude attached to the features, respectively. Here represents the bias of the model, and b represents the intercept.

Linear regression models try to optimize the β0 and b to minimize the cost function. The equation for the cost function for the linear model is given below:

$$\sum_{i=1}^{M}\left(y_i - y'_i\right)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{n} \beta_j * Xij\right)^2$$

Now, we will add a loss function and optimize parameter to make the model that can predict the accurate value of Y. The loss function for the linear regression is called as RSS or Residual sum of squares.

In regularization, we introduce a penalty term in the cost function to shrink the learned parameters to zero relative to the least square estimates and discourage using a complex model. The exact penalty term introduced depends on the regularization technique employed.

Here are some popular regularization techniques:

- **L1 regularization:** also known as Lasso regularization, adds a penalty proportional to the absolute value of the model weights to the loss function. This encourages the model to produce sparse solutions, where some weights are exactly zero.
- **L2 regularization:** also known as Ridge regularization, adds a penalty proportional to the square of the model weights to the loss function. This encourages the model to produce small but non-zero weights.
- **Elastic Net regularization:** combines both L1 and L2 regularization, by adding a penalty term that is a linear combination of the L1 and L2 penalties. This allows the model to learn sparse features while also maintaining some degree of regularization on all features.
- **Dropout:** is a technique that randomly drops out some neurons during training, which forces the model to learn redundant representations of the data. This reduces the risk of overfitting by preventing any one neuron from dominating the model.
- **Early stopping:** involves monitoring the performance of the model on a validation set during training and stopping training when the performance on the validation set stops improving. This prevents the model from continuing to learn the training data too well and overfitting.
- **Data augmentation:** involves creating new training data by applying transformations to the existing data, such as rotating, flipping, or cropping images. This increases the amount of training data available and helps the model to learn more robust features.

These techniques can be used alone or in combination to regularize machine learning models and prevent overfitting.

14. Which particular algorithms are used for regularization?

Ans:-There are three algorithms, are used for regularization :-

1. Lasso Regression (L1 Form)

Lasso regression (short for "Least Absolute Shrinkage and Selection Operator") is a type of linear regression that is used for feature selection and regularization. Adding a penalty term to the cost function of the linear regression model is a technique used to prevent overfitting. This encourages the model to use fewer variables or features in the final regression equation. The penalty term is based on the absolute values of the coefficients in the equation, which means that some of the coefficients may be set to zero if they are deemed to be less important.

The lasso regression model can be particularly useful when dealing with high-dimensional datasets, where the number of variables or features is much larger than the number of observations. In these cases, traditional linear regression models may suffer from overfitting, where the model is too complex and fits the training data too closely, resulting in poor performance on new, unseen data. Lasso regression can help to address this problem by identifying the most important variables and reducing the complexity of the model.

Below is an example of implementing L1 regularization using Python and scikit-learn on the California housing dataset

```
# Import required modules
from sklearn.linear_model import Lasso
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Load the California Housing dataset
X, y = fetch_california_housing(return_X_y=True)

# 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)

# Initialize the Lasso regression model with L1 regularization
model = Lasso(alpha=0.1)

# Fit the model to the training data
model.fit(X_train, y_train)

# Evaluate the model on the testing data
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print("MSE: ", mse)


# Output:
MSE:  0.6135115198058131
```

In this example, we load the California Housing dataset using the fetch_california_housing function from scikit-learn. We then split the data into training and testing sets using train_test_split. We initialize a Lasso regression model with an alpha value of 0.1, which controls the strength of the L1 regularization. We fit the model to the training data using the fit method, and evaluate the model on the testing data using predict and mean_squared_error.

2. RIDGE regression (L2 FORM) :-

Ridge Regression is a type of regularized linear regression that helps to prevent overfitting. It is similar to ordinary least squares regression, but with an additional penalty term added to the cost function. The penalty term is a regularization parameter, denoted by ‘alpha’, that controls the degree of shrinkage of the regression coefficients towards zero. The higher the value of alpha, the greater the degree of shrinkage and the more robust the model becomes. The value of α controls the strength of this penalty term and can be adjusted to obtain the best model performance on the validation set.

L2 form Penalizes the model based on the sum of squares of magnitude of the coefficients . It will shrink the coefficients for those predictors which contribute very less in the model but have huge weights, very close to zero but it never makes them exactly zero. Thus, the final model will still contain all those predictors, though with less weights.

In order to implement Ridge Regression in Python, we can use the Ridge module from the sklearn.linear_model library.

```
# Imports
from sklearn.linear_model import Ridge
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Load the Boston Housing dataset
boston = load_boston()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, test_size=0.3,
random_state=42)

# Instantiate the Ridge Regression model with alpha = 0.1
ridge = Ridge(alpha=0.1)

# Fit the model to the training data
ridge.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = ridge.predict(X_test)

# Calculate the mean squared error of the predictions
mse = mean_squared_error(y_test, y_pred)

# Print the mean squared error
print("Mean Squared Error:", mse)


## Output
Mean Squared Error: 21.5851159150243
```

In this example, we first loaded the Boston Housing dataset and split it into training and testing sets. We then instantiated the Ridge Regression model with an alpha value of 0.1 and fit the model to the training data. We made predictions on the testing data and calculated the mean squared error of the predictions. Finally, we printed the mean squared error to evaluate the performance of the Ridge Regression model.

Ridge regression is used when it is important to consider all the independent variables in the model or when many interactions are present. That is where collinearity or codependency is present amongst the variables.

3. ELASTIC NET

Elastic Net - The Elastic Net Regression technique is a combination of the Ridge and Lasso regression technique. It is the linear combination of penalties for both the L1-norm and L2-norm regularization. The model using elastic net regression allows the learning of the sparse model where some of the points are zero, similar to Lasso regularization, and yet maintains the Ridge regression properties. Therefore, the model is trained on both the L1 and L2 norms.

When many variables are present, and we can’t determine whether to use Ridge or Lasso regression, then the Elastic-Net regression is your safe bet.

15. Explain the term error present in linear regression equation?

Ans:

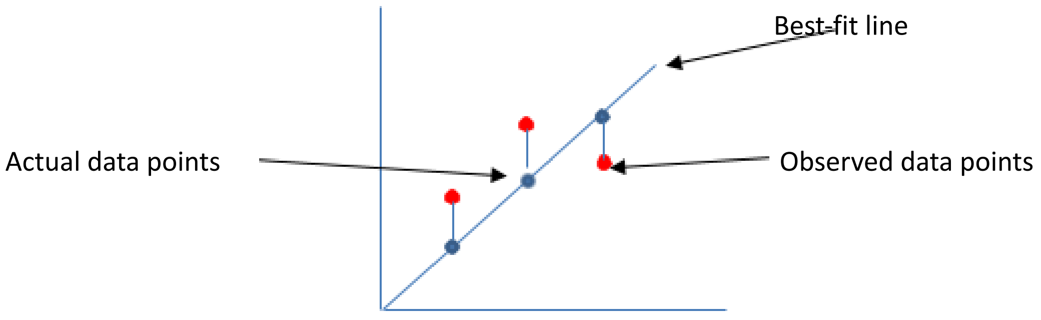
In linear regression, the "error" or "residual" refers to the difference between the predicted value and the actual value of the dependent variable.

The linear regression equation can be written as:

y = mx + b + e
where y is the dependent variable (or response variable),
x is the independent variable (or predictor variable),
m is the slope of the regression line,
b is the intercept,
and e is the error term.

The error term (e) represents the variability or randomness in the dependent variable that is not explained by the independent variable. In other words, it is the difference between the actual value of the dependent variable and the value predicted by the regression line.

It is the distance between each point and the linear graph is our **error term** (In our graph space between red dots and blue dots).



The goal of linear regression is to minimize the sum of the squared errors, which is known as the residual sum of squares (RSS). This is done by finding the values of m and b that minimize the distance between the predicted values and the actual values of the dependent variable. The regression line that results from this process is the "best-fit" line that represents the relationship between the independent and dependent variables.

Mean-Absolute Error (MSE)

The Mean Absolute Error (MAE) is the simplest regression error metric. It represents the average absolute difference between the actual and predicted values in the dataset. And measures the average of the residuals in the dataset.

```
# actual values
actual_values = [5,6,4,5,7,4]
# predicted avlues
predicted_values = [6,6,5,5,7,4.3]
# Importing mean_absolute_error from sklearn module
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
# printing the mean absolute error
print(mean_absolute_error(actual_values, predicted_values))
```

```
Output: 0.383333333333
```

Mean-Squared Error (MSE)

Linear Regression most often uses mean-square error (MSE) to calculate the error of the model. MSE is calculated by:

1. measuring the distance of the observed y-values from the predicted y-values at each value of x;
2. squaring each of these distances;
3. calculating the mean of each of the squared distances.

Linear regression fits a line to the data by finding the regression coefficient that results in the smallest MSE. The Mean Square Error (MSE) is similar to the Mean Absolute Error (MAE), but instead of taking the absolute value, it squares the difference before adding them together. The MSE will always be larger than the MAE because we squared the difference.

```
# actual values
actual_values = [5,6,4,5,7,4]
# predicted avlues
predicted_values = [3,6,5,5,7,4.3]
# Importing mean_absolute_error from sklearn module
from sklearn.metrics import mean_squared_error, mean_absolute_error
# printing the mean squared error
print(mean_squared_error(actual_values, predicted_values))
```

```
Output: 0.8483333333
```

The MSE is useful for ensuring that our trained model does not contain any outlier predictions with significant errors, as the squaring element of the function gives these errors more weight. However, if our model makes a bad prediction, the squaring part of the function magnifies the error.

Root-Mean-Squared Error (RMSE)

It is similar to MSE, but RMSE is the square root of the average squared difference between the predicted and actual value. The only problem with MSE is that the loss order is higher than the data order. As a result, we can’t simply correlate data to the error. While in RMSE, we are not changing the loss function, and the solution is still the same. All we are doing is reducing the order of the loss function by taking root.

```
# importing the square root method
from math import sqrt
# actual values
actual_values = [5,6,4,5,7,4]
# predicted avlues
predicted_values = [3,6,5,5,7,4.3]
# Importing mean_absolute_error from sklearn module
from sklearn.metrics import mean_squared_error
# printing the root mean absolute error
print(sqrt(mean_squared_error(actual_values, predicted_values)))
```

```
Output: 0.9210501253098733
```

R-Squared Error

The R-Squared Error method is also known as the coefficient of determination. This metric indicates how well a model fits a given dataset. Or in simple words, it indicates how close the regression line is to the actual data values. The R-Squared value lies between 0 and 1, where 0 indicates that this model doesn't fit the given data and 1 means that the model perfectly fits the dataset provided.

```
# actual values
actual_values = [5,6,4,5,7,4]
# predicted avlues
predicted_values = [3,6,5,5,7,4.3]
# importing r square error
from sklearn.metrics import r2_score
# applying r square error
R_square = r2_score(actual_values, predicted_values)
print(R_square)
```

```
Output: 0.2551219512195122
```