M.Tech Program

Advanced Industry Integrated Programs

Jointly offered by University and LTIMindTree

Python for Data Science

Knowledge partner

LTIMindtree

Implementation partner

L&T EduTech

Modules to cover

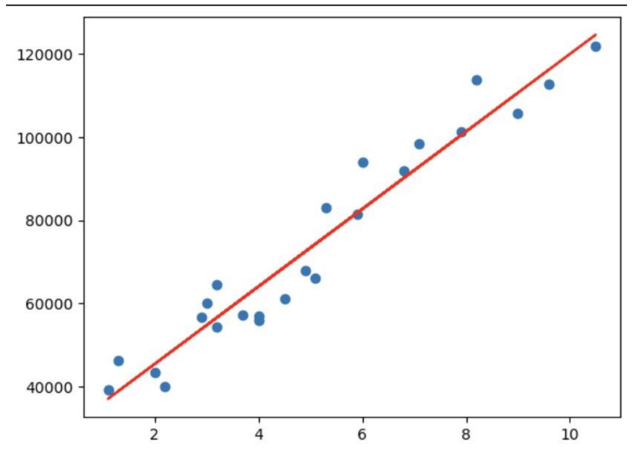
1. Python - Data Structures, OOPS & Modules
2. Python - Numpy, Pandas & DS Libraries
3. Visualization
4. **Regression and Classification**
5. Unsupervised and Advanced Machine Learning

Regression and Classification

Regression and Classification

Introduction to Linear Regression

- Linear regression is a supervised machine learning algorithm that identifies the linear relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data.
- When there is only one independent variable, it is known as simple linear regression. With multiple independent variables, it is called multiple linear regression.



Regression and Classification

Introduction to Linear Regression

Simple Linear Regression

This is the most basic form of linear regression, involving just one independent variable and one dependent variable. The equation for simple linear regression is: :

$$y = \beta o + \beta_1 X$$

Where, **Y** is the dependent variable

β0 is the intercept

β1 is the slope

X is the independent variable

Regression and Classification

Introduction to Linear Regression

Evaluation metrics

Mean Absolute Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \widehat{Y}_i\right|$$

Mean Squared Error(MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2$$

Root Mean Squared Error(RMSE)

$$RMSE = sqrt(\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2)$$

R Squared Error(R2)

$$R^2 = 1 - \frac{\sum(y_i - \widehat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Regression and Classification

Cost Functions

- In linear regression, you're trying to fit a straight line to your training data. This line is represented by the equation **f(x) = a * x + b**, where w and b are the model's parameters.
- The cost function, denoted by J(a, b), measures how well a particular choice of w and b fits the training data.
- The goal of linear regression is to find the values of w and b that minimize the cost function J(a, b), making the line fit the data points as closely as possible.

Regression and Classification

Cost Functions

- The most common cost function used in linear regression is the mean squared error cost function:

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y_i)^2$$

where m is the number of training examples

$\widehat{y_i}$ is the predicted value for the ith data point

y_i is the actual y value for the ith data point in your training set.

Regression and Classification

Cost Functions

1

Guides the Learning Algorithm

The cost function acts as a guide for the learning algorithm in linear regression.

2

Minimizes Squared Errors

By minimizing the cost function, you essentially minimize the overall squared errors between the predicted values (f(x)) and the actual y values.

3

Indicates Better Fit

A lower cost function indicates a better fit between the line and the data points.

Regression and Classification

Gradient Descent

- Gradient descent is an iterative optimization algorithm used in training machine learning models. It aims to find the parameter values of a function that minimize the cost function as much as possible.
- Repeat Until convergence {

$$a = a - \alpha \frac{\partial}{\partial a} J(a, b)$$

$$b = b - \alpha \frac{\partial}{\partial a} J(a, b)$$

}

Regression and Classification

Gradient Descent

starts with an initial guess for the parameters of the model (weights and biases)

calculates the gradient of the cost function with respect to the parameters.

The gradient points in the direction of the steepest ascent of the cost function

Gradient descent takes a small step in the opposite direction (downhill direction), which is the direction of steepest descent.

This process is repeated until the cost function converges to a minimum value

Regression and Classification

Simple Python coding

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([[1], [2], [3], [4], [5]])
y = np.array([1, 3, 3, 2, 5])

model = LinearRegression() #Syntax
model.fit(X, y)

print(model.intercept_)
print(model.coef_)

X_new = np.array([[6], [7]])
predictions = model.predict(X_new)
print(predictions)
```

Regression and Classification

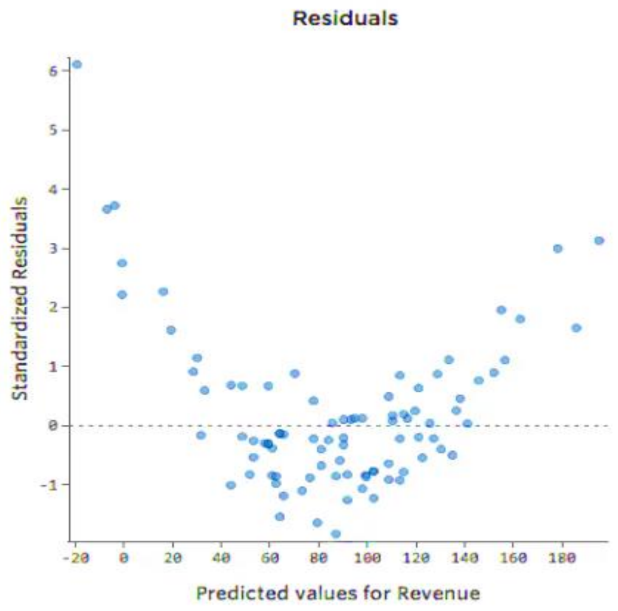
Overview of Scikit-Learn and Python

- Scikit-Learn is a robust Python library for machine learning.
- Offers straightforward and efficient tools for data analysis and modeling.
- Contains modules for preprocessing, classification, regression, clustering, and additional tasks.
- Seamlessly integrates with other libraries such as NumPy and Pandas.
- Provides comprehensive documentation and community support.

Regression and Classification

Residual Plots

- Residual plots show the difference between predicted and actual values.
- Useful for diagnosing the fit of a regression model.
- Ideally, residuals should be randomly distributed with no clear pattern.
- Patterns in residual plots indicate issues like non-linearity or heteroscedasticity.
- Helps in validating the assumptions of linear regression.



Regression and Classification

Model Deployment and Coefficient Interpretation

- Deploying a regression model involves making it available for use in production.
- Coefficients represent the relationship between independent variables and the dependent variable.
- Positive coefficients indicate a direct relationship; negative coefficients indicate an inverse relationship.
- The magnitude of coefficients shows the strength of the relationship.
- Interpretation of coefficients is critical for understanding the model's insights.

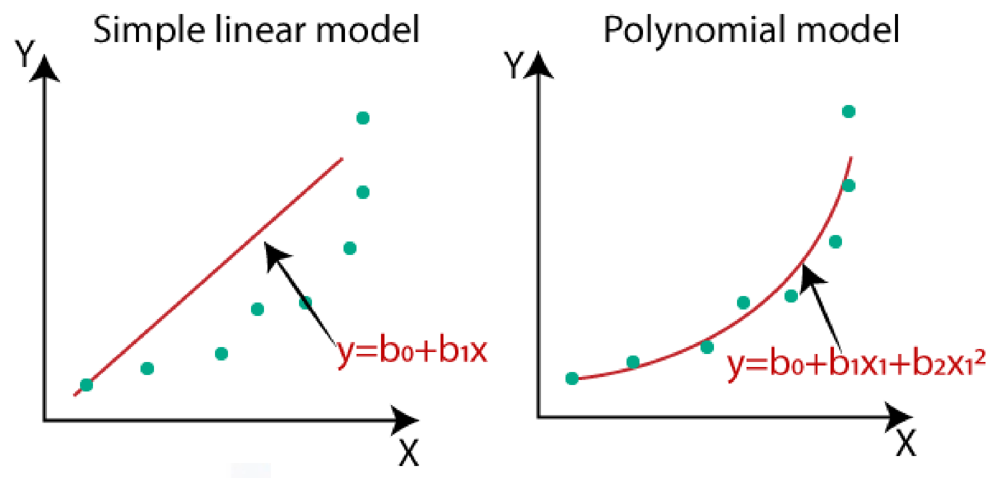
Polynomial Regression

Regression and Classification

Polynomial Regression

- Some curvy data can be modeled by a polynomial regression
- For example: $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

A polynomial regression model can be transformed into linear regression model



$x_1 = x$ $x_2 = x^2$ $x_3 = x^3$

$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$ → Multiple linear regression → Least Squares

Minimizing the sum of the squares of the differences between y and $\hat{y}$

Regression and Classification

Polynomial Regression

- To capture a non-linear relationship between the dependent variable and a set of independent variables
- $\hat{Y}$ must be a non-linear function of the parameters θ, rather than the features x.

$$\hat{y} = \theta_0 + {\theta_2}^2 x$$

$$\hat{y} = \theta_0 + \theta_1 {\theta_2}^x$$

$$\hat{y} = \log(\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3)$$

$$\hat{y} = \frac{\theta_0}{1 + \theta^{(x-\theta_2)}}$$

Regression and Classification

Polynomial Regression

Linear vs non-linear regression

How can I easily determine if a problem is linear or non-linear?

- Visually inspect the data.
- Assess based on accuracy.

How should I model my data if it appears non-linear on a scatter plot?

- Use polynomial regression.
- Apply a non-linear regression model.
- Transform your data.

Regression and Classification

Polynomial Regression

Demo:

Exploring and visualizing the dataset

Preprocessing data to fit ML models

Building and training a Polynomial regression

Calculating accuracy

Regression and Classification

Polynomial Regression -Theory and Motivation

- Polynomial regression is motivated by the need to model non-linear relationships.
- It enhances the flexibility of the regression model.
- Allows the capture of curved trends in the data.
- Provides better fit for data that does not follow a linear pattern.
- However, it comes with increased risk of overfitting.

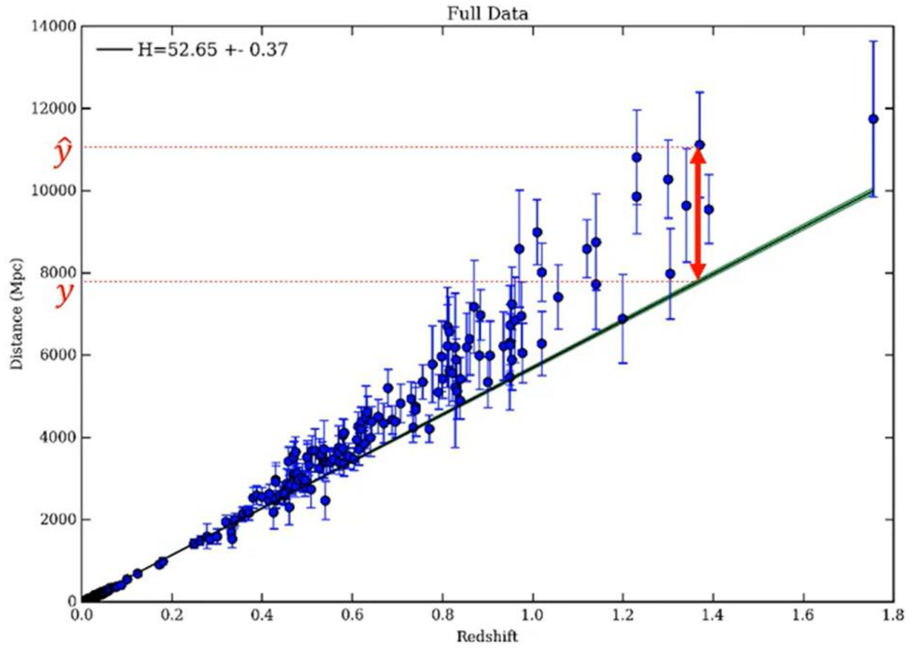
Regression and Classification

Creating Polynomial Features

- Polynomial features are created by raising original features to a power.
- Example: For a feature x, create x^2, x^3 etc.
- Use Scikit-Learn's Polynomial Features module to automate this process.
- These features are then used in the regression model.
- Careful selection of polynomial degree is crucial.

Regression and Classification

Training and Evaluation



$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2}$$

$$RAE = \frac{\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right|}{\sum_{j=1}^{n}\left|y_j - \bar{y}\right|}$$

$$RSE = \frac{\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2}{\sum_{j=1}^{n}\left(y_j - \bar{y}\right)^2}$$

$$R^2 = 1 - RSE$$

Regression and Classification

Bias Variance Trade Off

- Bias: Error due to overly simplistic assumptions in the model.
- Variance: Error caused by excessive complexity and sensitivity to data fluctuations.
- A good model balances bias and variance for optimal performance.
- High bias leads to underfitting; high variance leads to overfitting.
- Techniques like cross-validation and regularization help manage the bias-variance trade-off effectively.

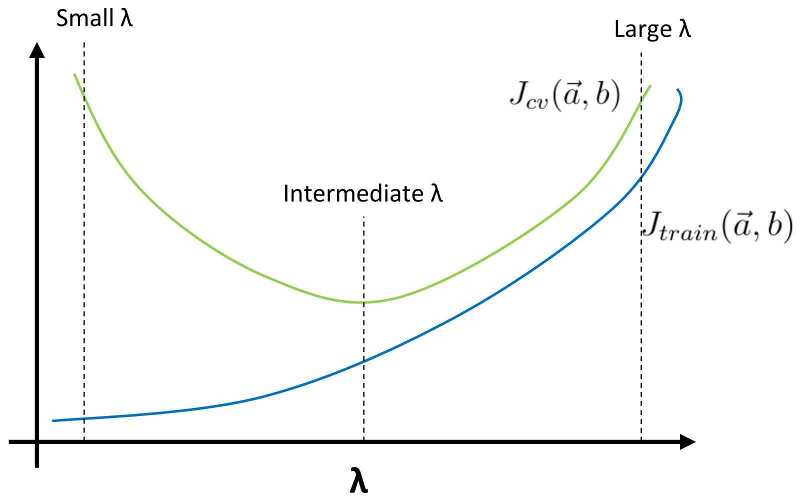
Regression and Classification

Bias Variance Trade Off

- Consider,

$$J(\vec{a},b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{\vec{a},b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} a_j^2$$

- Choose an intermediate value for λ, where the cross-validation error is low, this would yield a good model for application.



Regression and Classification

Choosing Degree of Polynomial

- The degree of the polynomial determines the model's complexity.
- Higher degrees can capture more intricate patterns but risk overfitting.
- Use cross-validation to choose the optimal degree.
- Plot learning curves to visualize model performance against polynomial degree.
- Aim for a balance between model accuracy and complexity.

Regression and Classification

Model Deployment

- Train Model: Develop the polynomial regression model using historical data to uncover patterns.
- Save Model: Store the learned model parameters in a file for later use.
- Deployment Pipeline: Create a pipeline to load the model and preprocess new data for predictions.
- API Integration: Integrate the model into an API to enable real-time predictions from external systems.
- Monitor and Maintain: Regularly monitor model performance and make updates as necessary to maintain accuracy and usefulness.

Regression and Classification

Feature Scaling

- Feature scaling ensures that all features contribute equally to the model.
- Common scaling techniques: Standardization and Min-Max Scaling.
- Standardization scales features to have zero mean and unit variance.
- Min-Max Scaling scales features to a fixed range, usually [0, 1].
- Scaling is critical for algorithms like Gradient Descent to converge properly.

Regression and Classification

Introduction to Cross Validation

Use of Cross Validation

- Prevents overfitting.
- Provides insight into the model's performance across different subsets of the data.
- Ensures the model's reliability and robustness.

Types of Cross Validation

- K-Fold Cross validation
- Leave-One-Out Cross validation
- Stratified K-Fold Cross validation

Regression and Classification

Introduction to Cross Validation

Types of Cross Validation

- **K-Fold Cross Validation:** Splits data into K subsets, trains on K-1, tests on the remaining fold, iterates K times.
- **Leave-One-Out Cross Validation (LOOCV):** Each data point is used as a single test instance, training on the remaining data.
- **Stratified K-Fold Cross Validation:** Ensures each fold has the same class distribution as the full dataset, improving reliability for imbalanced data.

Regression and Classification

Regularization Data Setup

- **Standardization:** Ensure data is standardized (mean=0, variance=1) for optimal performance of regularization techniques.
- **Feature Scaling:** Important to scale features to avoid dominance of features with larger scales.
- **Training and Validation Split:** Use a separate validation set to tune regularization parameters.
- **Hyperparameter Tuning:** Use techniques like cross-validation to find optimal regularization strength (λ).

Regression and Classification

Regularization Data Setup

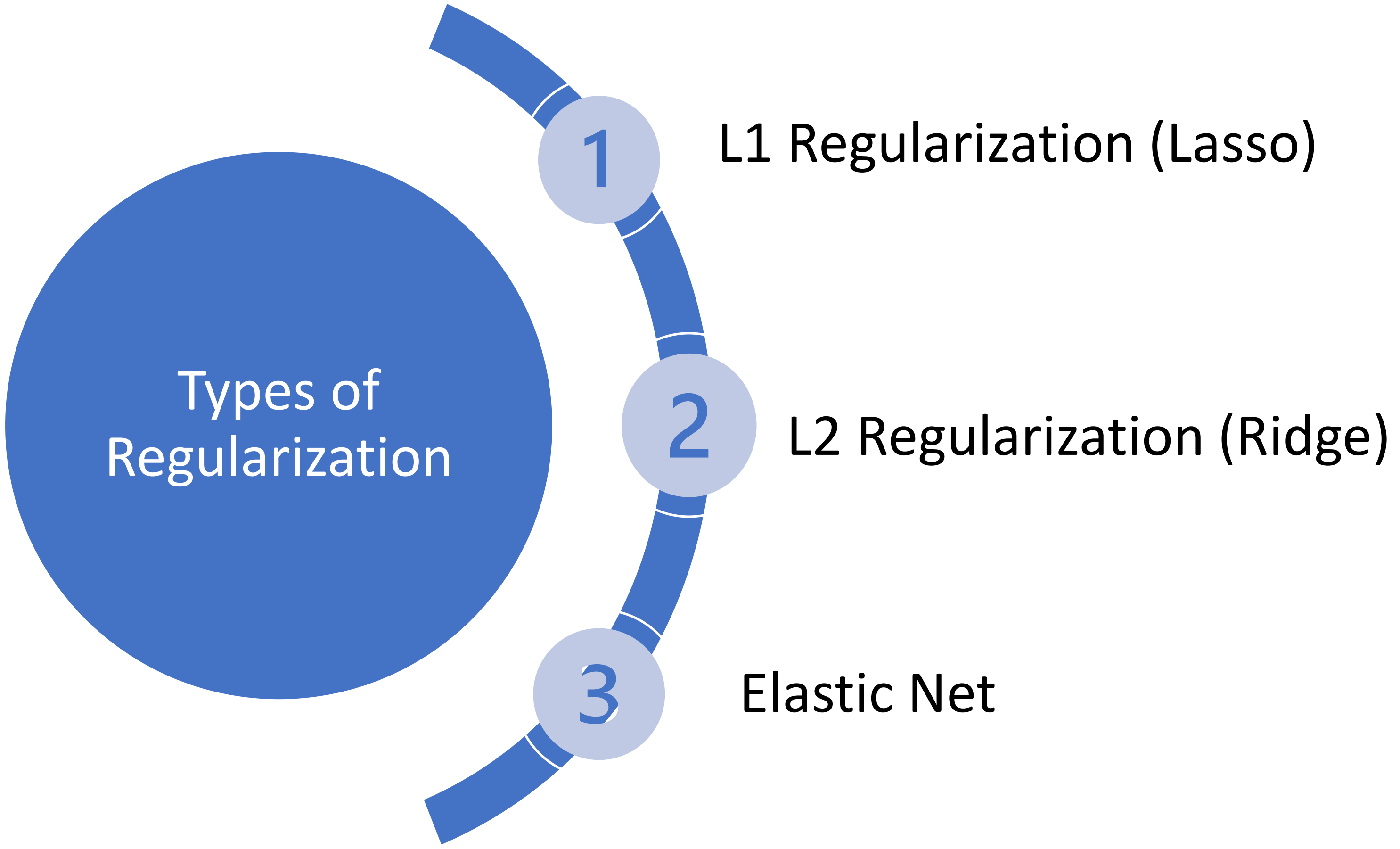
Implementation

- **Standard Libraries:** Use libraries like scikit-learn, which have built-in support for regularization.
- **Parameter Selection:** Start with a wide range of regularization parameters and narrow down using grid search or random search.
- **Model Evaluation:** Assess model performance on validation data to ensure effective regularization.

Regression and Classification

Regularization

Types of Regularization



Regression and Classification

Ridge Regression Theory

Purpose

- Prevents overfitting by penalizing large coefficients.
- Improves model generalization to new data.

Mathematical Formula

$$||y - X\beta||^2 + \lambda \sum |\beta|^2$$

where y is the target vector, X is the feature matrix, β is the coefficient vector, and λ is the regularization parameter.

Regression and Classification

Ridge Regression Theory

Working

- Adds a penalty equal to the sum of the squared coefficients.
- Controls the complexity of the model by shrinking the coefficients.
- Larger λ values lead to greater shrinkage of coefficients, while λ=0 results in ordinary least squares (OLS) regression.

Key Properties

- Bias-Variance tradeoff
- Numerical stability

Regression and Classification

Lasso Regression

Purpose

- Prevents overfitting by adding a penalty to the model for large coefficients.
- Performs feature selection by driving some coefficients to exactly zero.

Mathematical Formula

$$||y - X\beta||^2 + \lambda\sum|\beta|$$

where y is the target vector, X is the feature matrix, β is the coefficient vector, and λ is the regularization parameter.

Regression and Classification

Lasso Regression

Working

- Adds a penalty equal to the absolute value of the coefficients.
- Shrinks some coefficients to exactly zero, effectively performing feature selection.
- Larger λ values increase the shrinkage effect, leading to sparser models.

Properties

- Feature Selection
- Bias-Variance Tradeoff
- Sparse Solutions

Regression and Classification

Lasso Regression- Background and Implementation

1. **Data Preparation:**
   - Standardize features to have zero mean and unit variance.
   - Split the data into training and testing sets.
2. **Model Training:**
   - Choose a range of λ values for hyperparameter tuning.
   - Fit the Lasso regression model using training data for each λ.

```
from sklearn.linear_model import Lasso
final_model = Lasso(alpha=best_alpha)
final_model.fit(X_train, y_train)
```

Regression and Classification

Lasso Regression- Background and Implementation

4. **Cross-Validation :**

   Use cross-validation techniques to evaluate model performance and select the optimal λ value.

5. **Model Evaluation :**

   Assess the model's performance on the test dataset using appropriate evaluation metrics

test_score = final_model.score(X_test, y_test)

Regression and Classification

Difference between Lasso and Ridge Regression

	Lasso Regression	Ridge Regression
Regularization Type	Uses L1 regularization	Uses L2 regularization
Coefficient Shrinkage	shrinks coefficients exactly zero	Shrinks coefficients towards zero
Model Complexity	Tends to produce simpler, more interpretable models	Tends to produce models that include all features
Handling Multicollinearity	May choose one feature among highly correlated ones, potentially ignoring others.	Distributes the effect across correlated features, handling multicollinearity better.
Use Cases	Best when there are many irrelevant features	Best when dealing with multicollinearity
Mathematical formula	$\|\|y - X\beta\|\|^2 + \lambda\sum\|\beta\|$	$\|\|y - X\beta\|\|^2 + \lambda\sum\|\beta\|^2$

Regression and Classification

Elastic Net

Purpose

- Provides a balance between feature selection (Lasso) and coefficient shrinkage (Ridge).
- Effective for datasets with highly correlated features.

Mathematical Formula

$$||y - X\beta||^2 + \lambda_1 \sum |\beta|^2 + \lambda_2 \sum |\beta|$$

where y is the target vector, X is the feature matrix, β is the coefficient vector, and λ_1 controls the L1 penalty, and λ_2 controls the L2 penalty.

Regression and Classification

Elastic Net

Key Properties

- **Combination of Penalties:**

  Uses both L1 and L2 penalties to regularize the model.

- **Feature Selection:**

  Retains Lasso's ability to select features by setting some coefficients to zero.

- **Handling Multicollinearity:**

  Mitigates issues with correlated features better than Lasso.

Regression and Classification

Feature Engineering and Data Preparation

Feature Engineering

- The process of creating new features or modifying existing ones to improve model performance.
- Involves transforming raw data into meaningful features that better represent the underlying problem.

Importance:

- Enhances model accuracy and predictive power.
- Helps in identifying and incorporating relevant information.
- Can lead to simpler, more interpretable models.

Regression and Classification

Feature Engineering and Data Preparation

Steps in Feature Engineering

Steps	
Handling Missing Data	• **Imputation:** Fill missing values using strategies like mean, median, mode, or advanced methods (e.g., KNN imputation). • **Removal:** Drop rows or columns with excessive missing values.
Encoding Categorical Variables	• **One-Hot Encoding:** Converts categorical values into binary vectors. • **Label Encoding:** Assigns a unique integer to each category.
Scaling and Normalization	• **Standardization:** Rescales features to have zero mean and unit variance. • **Normalization:** Scales features to a range (e.g., 0 to 1) to ensure equal importance.

Regression and Classification

Feature Engineering and Data Preparation

Steps in Feature Engineering

Steps	
Feature Creation	• **Polynomial Features:** Generates interaction terms and polynomial combinations of features. • **Domain-Specific Features:** Create features based on domain knowledge (e.g., ratios, differences).
Dimensionality Reduction	• **PCA (Principal Component Analysis):** Reduces dimensionality while retaining most variance. • **LDA (Linear Discriminant Analysis):** Reduces dimensions while maximizing class separability.

Regression and Classification

Feature Engineering and Data Preparation

Steps in Feature Engineering

Steps	
Feature Selection	• **Filter Methods:** Select features based on statistical tests (e.g., correlation, chi-square). • **Wrapper Methods:** Use algorithms (e.g., RFE) to select features based on model performance. • **Embedded Methods:** Integrate feature selection during model training (e.g., Lasso).

Regression and Classification

Feature Engineering and Data Preparation

Data Preparation

- Involves removing errors, handling missing values, normalizing, and standardizing data to ensure accuracy and consistency.
- Creating and modifying features to enhance model performance and dividing the dataset into training, validation, and test sets for unbiased evaluation.

Importance:

- Clean, well-prepared data leads to more accurate and reliable models.
- Properly scaled and transformed data ensures algorithms perform optimally.
- Prevents models from learning noise and irrelevant details, reducing overfitting.
- Streamlines the analysis process, saving time and resources by avoiding costly errors later.

Regression and Classification

Feature Engineering and Data Preparation

Steps in Data preparation

Steps	
Data Cleaning	• Detect and correct errors or inconsistencies in the dataset. • Handle outliers using methods like capping or removal.
Data Integration	• Combine data from different sources to create a unified dataset. • Ensure consistency and resolve conflicts in data formats.

Regression and Classification

Feature Engineering and Data Preparation

Steps in Data preparation

Steps	
Data Transformation	• Apply mathematical transformations (e.g., log, square root) to stabilize variance or normalize distributions. • Binning continuous variables into categorical ones based on thresholds.
Splitting the Data	• **Training and Testing Sets:** Divide data into training and testing subsets to evaluate model performance. • **Validation Set:** Use a separate validation set for hyperparameter tuning and model selection.

Regression and Classification

Dealing with outliers

Identifying Outliers:

1. **Statistical methods:**

   a) **Z-score:** $Z = \frac{(X - \mu)}{\sigma} \longrightarrow$ Typically, data points with a Z-score > 3 or < -3 are considered outliers.

   b) **Interquartile Range (IQR) Method:**

$$IQR = Q3 - Q1$$

$$\boldsymbol{Lower\ bound} = Q1 - 1.5 * IQR \qquad \boldsymbol{Upper\ bound} = Q3 + 1.5 * IQR$$

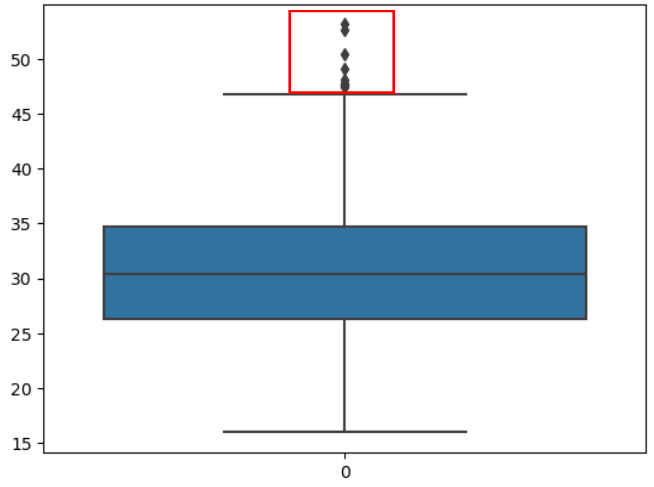
Data points outside this range are considered outliers.

Regression and Classification

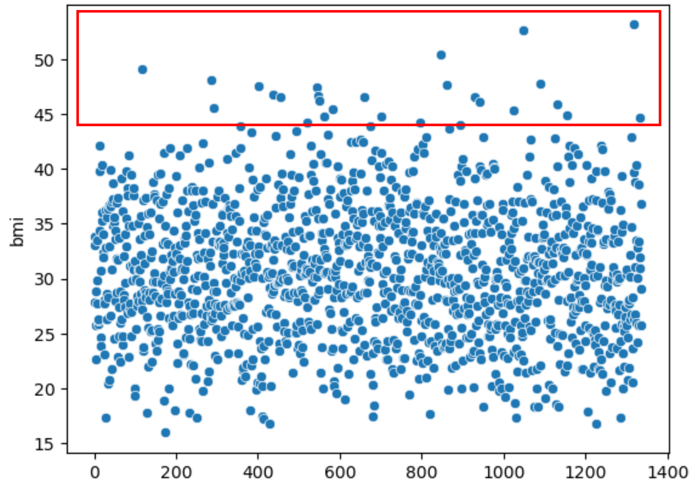
Dealing with outliers

Identifying Outliers:

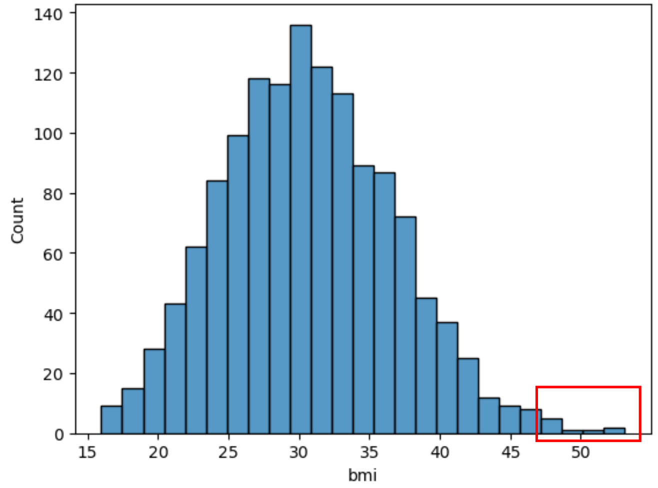
2. **Visualization methods :**



Box Plot



Scatter plot



Histogram

Regression and Classification

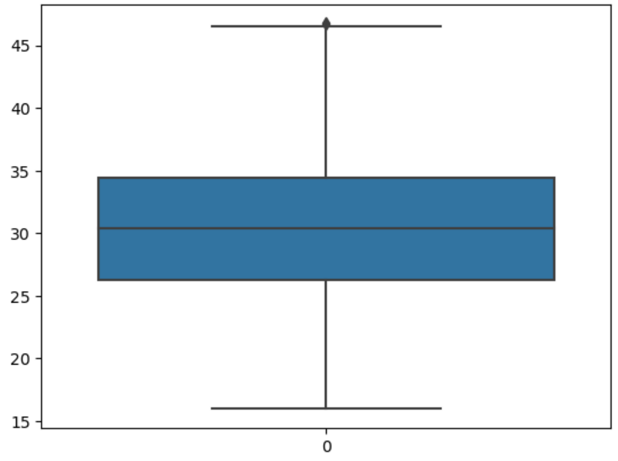
Dealing with outliers

Handling Outliers:

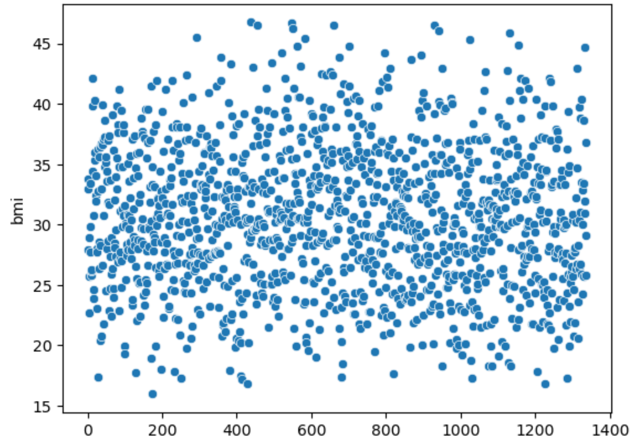
- Most effective Way to handle the outlier is to remove the outlier.

```
df_clean = df[(df['X'] >= lower_bound) & (df['X'] <= upper_bound)]
```

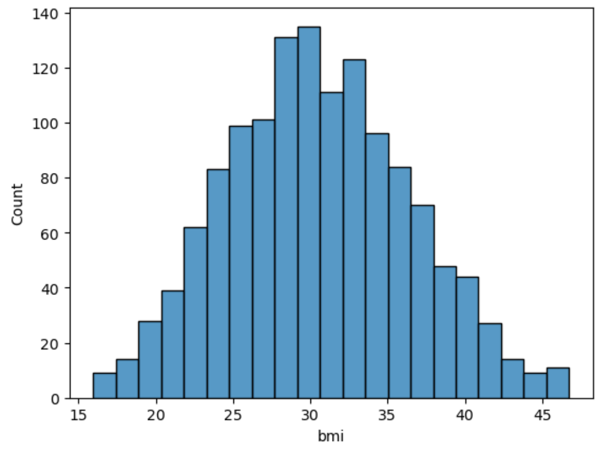
- Data visualization after removing the outliers.



Box Plot



Scatter plot



Histogram

Regression and Classification

Dealing with Missing Data

Identifying presence of missing data:

df.isnull().any() →	#returns a boolean Series indicating whether each column in the DataFrame has any missing values
dataset.isnull().sum() →	#returns a Series that contains the count of missing values in each column of the DataFrame.

Regression and Classification

Evaluation of Missing Data

Why Evaluate Missing Data?

- Understanding the pattern and mechanism of missing data is crucial for choosing the right method to handle it.
- Different patterns of missing data require different strategies for imputation or handling.

Percentage of Missing Data

- A high percentage of missing data in a column may indicate the need to drop that column or use advanced imputation methods.

```
missing_percentage = df.isnull().mean() * 100
```

Regression and Classification

Evaluation of Missing Data

Patterns of missing Data

- Missing Completely at Random (MCAR)

  The missingness is unrelated to any data, observable or unobservable.

- Missing at Random (MAR)

  The missingness is related to observed data but not the missing data itself.

- Missing Not at Random (MNAR)

  The missingness is related to the value of the missing data itself.

Regression and Classification

Filling or Dropping Data Based on Rows

Removing Rows with Missing Values

If only a small number of rows have missing values, you might simply drop these rows.

```
df_dropna_rows = df.dropna()
```

Imputation Methods

- **Mean Imputation**

```
df_mean_impute = df.fillna(df.mean())
```

- **Median Imputation**

```
df_median_impute = df.fillna(df.median())
```

Regression and Classification

Filling or Dropping Data Based on Rows

Imputation Methods

- Mode Imputation

```
df_mode_impute = df.fillna(df.mode().iloc[0])
```

- Forward Fill (propagating the last valid observation forward to next)

```
df_ffill = df.fillna(method='ffill’)
```

- Backward Fill (using the next valid observation to fill previous missing values)

```
df_bfill = df.fillna(method='bfill')
```

Regression and Classification

Fixing Data Based on Columns

When to Remove Columns with Missing Data

- If a column has a very high percentage of missing data (e.g., >50%), it might be more practical to remove it.
- Columns that don't contribute much to the analysis or are redundant can be dropped if they have missing values.

```
threshold = len(df) * 0.5 # Remove columns with more than 50% missing values
df_drop_columns = df.dropna(thresh=threshold, axis=1)
```

Regression and Classification

Fixing Data Based on Columns

Imputation for Specific Columns

- Use mean Imputation technique for numeric columns

```
df['col1'].fillna(df['col1'].mean(), inplace=True)
```

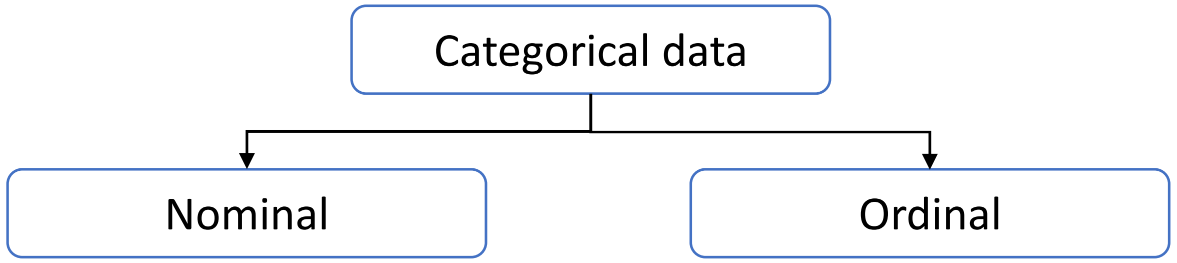
- Use mode imputation technique for categorical columns

```
df['col2'].fillna(df['col2'].mode()[0], inplace=True)
```

Regression and Classification

Dealing with Categorical Data

- Categorical data represent characteristics and can take on a limited, fixed number of possible values.



- Categorical data must be converted into a numerical format for machine learning algorithms.
- Missing values in categorical data need to be addressed carefully to avoid bias.

Regression and Classification

Encoding Options

Encoding transforms categorical data into numerical format, essential for machine learning algorithms which require numerical input. A few encoding methods are:

One-Hot Encoding

```
df_onehot = pd.get_dummies(df, columns=['col1'])
```

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Color'] = label_encoder.fit_transform(df['Color'])
```

Regression and Classification

Encoding Options

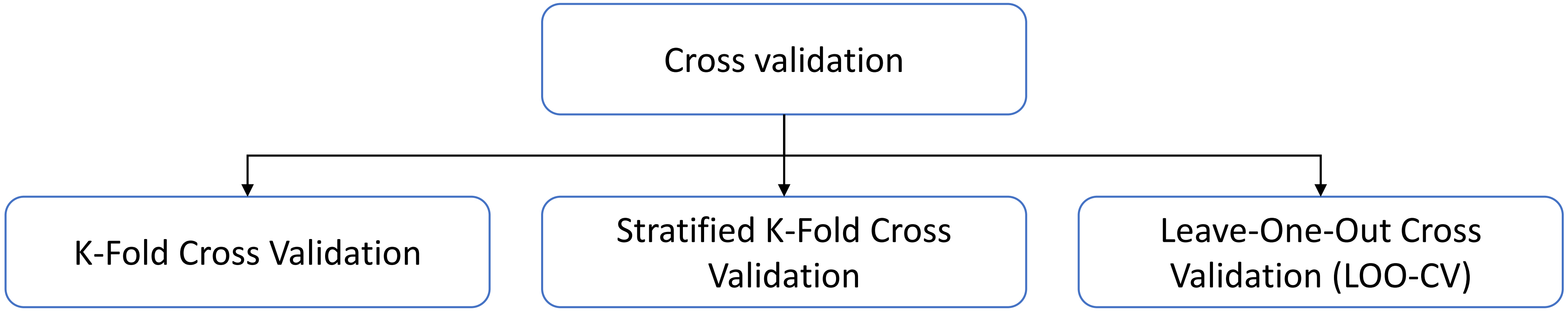
Ordinal Encoding

```
size_mapping = {        'Feature1': 1,
                        'Feature2': 2,
                        'Feature3': 3,
                        'Feature4': 4   }
df['col1'] = df['col1].map(size_mapping)
```

Regression and Classification

Cross Validation

- Cross validation is a technique for assessing how a machine learning model will generalize to an independent dataset.
- Helps in detecting overfitting and provides a better estimate of model performance.


Cross validation
K-Fold Cross Validation
Stratified K-Fold Cross Validation
Leave-One-Out Cross Validation (LOO-CV)

Regression and Classification

Cross Validation

K-Fold Cross Validation

The dataset is randomly partitioned into k equal-sized folds. Each fold is used once as a validation while the k-1 remaining folds form the training set.

```
from sklearn.model_selection import Kfold
kf = KFold(n_splits=5)
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Regression and Classification

Cross Validation

Stratified K-Fold Cross Validation

Ensures that each fold has the same proportion of classes as the original dataset.

```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=3)
for train_index, test_index in skf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

Regression and Classification

Cross Validation

Leave-One-Out Cross Validation (LOO-CV)

Uses a single observation as the validation set and the remaining observations as the training set.

```
from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()
for train_index, test_index in loo.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
```

Regression and Classification

Test-Validation-Train Split

- Data splitting involves dividing the dataset into distinct subsets to train, validate, and test a machine learning model.
- Usually the data is split into,

  Training Set: The largest portion of the data used to train the model. Typically, 60-80% of the data.

  Validation Set: Used to tune hyperparameters and evaluate the model during training. Typically, 10-20% of the data.

Regression and Classification

Test-Validation-Train Split

Test Set: Used to evaluate the final model's performance after training.

Typically, 10-20% of the data.

Splitting the data into training and testing set,

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Further splitting the training data into training and validation sets,

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=42)

Regression and Classification

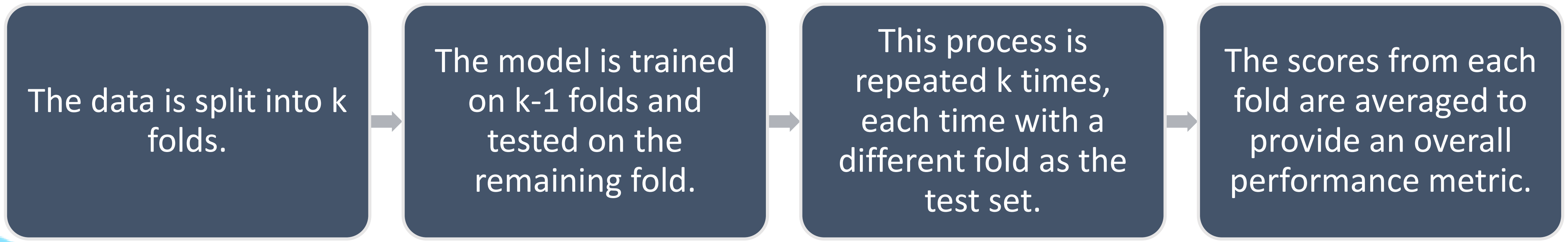
cross_val_score

- **cross_val_score** is a function from scikit-learn that performs cross-validation on a dataset and returns an array of scores for each fold.

  from sklearn.model_selection import cross_val_score

  scores = cross_val_score(model, X, y, cv=5)

- **How cross_val_score Works**



The data is split into k folds.

The model is trained on k-1 folds and tested on the remaining fold.

This process is repeated k times, each time with a different fold as the test set.

The scores from each fold are averaged to provide an overall performance metric.

Regression and Classification

cross_validate

- **cross_validate** is a function in scikit-learn that evaluates a model using cross-validation, providing multiple metrics and more detailed information compared to **cross_val_score.**

```
from sklearn.model_selection import cross_validate
cv_results = cross_validate(model, X, y, cv=5, scoring=('accuracy', 'f1_macro'))
```

- **Keys in cv_results:**

  test_score: Scores on the test set.

  train_score: Scores on the training set (if return_train_score=True).

  fit_time: Time taken to fit the model on each fold.

  score_time: Time taken to score the model on each fold.

Regression and Classification

Grid Search

Definition and Purpose

Grid Search:

A technique to find the optimal hyperparameters for a model by exhaustively searching through a specified subset of hyperparameter combinations.

Purpose:

Maximizes model performance by tuning hyperparameters to improve accuracy, robustness, and generalization.

Regression and Classification

Grid Search

Hyperparameter Tuning

Hyperparameters:

Parameters that are not directly learned by the model during training but control the learning process.

Tuning:

Adjusting hyperparameters to optimize model performance.

Regression and Classification

Grid Search

Implementation

GridSearchCV:

A class from scikit-learn that performs grid search with cross-validation.

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
grid_search = GridSearchCV(estimator=rf_classifier,
param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Regression and Classification

Linear Regression Project: The Logistic Function

- Logistic Regression predicts the probability of a binary outcome.
- Utilizes the logistic function to map predictions onto probabilities.
- The logistic function ensures outputs are constrained between 0 and 1.
- Ideal for solving binary classification tasks.
- The logistic function is defined as:

$$\sigma(z) = \frac{1}{1+e^{-z}}.$$

Regression and Classification

Logistic Regression

- Logistic regression is a supervised learning technique employed in binary classification tasks to predict the probability of an input belonging to a specific class.
- It uses a logistic function to establish the relationship between input features and the binary outcome, ensuring output values range between 0 and 1.

Regression and Classification

Logistic Regression

Advantages of Logistic Regression

1. **Simplicity and Interpretability:** Easy to implement and interpret, providing clear insights into the relationship between features and the outcome.
2. **Probability Estimates:** Outputs probabilities for class membership, useful for decision-making and risk assessment.
3. **Efficient:** Computationally efficient and works well with linearly separable data, making it suitable for large datasets.

Regression and Classification

Logistic Regression

Limitations of Logistic Regression

- **Linear Boundaries:** Assumes a linear relationship between the features and the log-odds of the outcome, which may not capture complex patterns.
- **Multicollinearity:** Performance can be negatively affected by high correlation among input features, leading to unreliable estimates.
- **Not Suitable for Non-Linear Problems:** Struggles with datasets where the decision boundary is non-linear, requiring feature engineering or alternative algorithms to improve accuracy.

Regression and Classification

Logistic Regression: Theory and Intuition

- Logistic Regression estimates the probability that a given input belongs to a class.
- Based on the logistic function, which provides a smooth transition between 0 and 1.
- The model outputs probabilities, which are then thresholded to make class predictions.
- Useful for binary classification tasks like spam detection, disease diagnosis.
- Provides interpretable coefficients indicating the impact of features

Regression and Classification

Linear to Logistic: Logistic Regression

- Logistic Regression extends linear regression to handle classification.
- Uses a logistic function to transform linear output into probabilities.
- Fits a logistic curve to the data, capturing the relationship between input features and the probability of the target class.
- Balances the trade-off between accuracy and interpretability.
- Handles binary outcomes effectively.

Regression and Classification

Linear to Logistic Math

- Linear regression equation: z=mx+b.
- Logistic regression employs the logistic function :

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad where\ Z = mx + b$$

- Converts the linear output into a probability ranging from 0 to 1.
- Utilizes Maximum Likelihood Estimation (MLE) to estimate parameters.
- Aims to maximize the likelihood of the observed data.

Regression and Classification

Logistic Regression - Model Training

- Train the logistic regression model by splitting the dataset into training and testing sets.
- Fit the model on the training data using the `fit` method in Scikit-Learn.
- Use the trained model to make predictions on the test data with the `predict` method.
- Evaluate model performance using classification metrics such as accuracy, precision, and recall.

Regression and Classification

Logistic Regression Model Training: Classification Metrics

- Classification metrics evaluate the performance of classification models.
- Common metrics: Accuracy, Precision, Recall, F1 Score, ROC AUC.
- Accuracy: Proportion of correct predictions.
- Precision: Proportion of true positive predictions.
- Recall: Proportion of actual positives correctly predicted.

Regression and Classification

Confusion Matrix and Accuracy

- Confusion matrix illustrates how well a classification model performs.
- Displays the counts of true positives, true negatives, false positives, and false negatives.
- Accuracy is calculated as:

$$\frac{TP+TN}{TP+TN+FP+FN}.$$

- Aids in comprehending the various types of classification mistakes.
- Particularly valuable for datasets with uneven class distributions.

Regression and Classification

Precision, Recall, F1 Score

- Precision: Proportion of correctly predicted positive instances among all predicted positive instances. $\frac{TP}{TP + FP}$
- Recall: Proportion of actual positive instances correctly predicted by the model. $\frac{TP}{TP + FN}$
- F1 Score: Harmonic mean of Precision and Recall, balancing their trade-off.

$$\frac{Precision. Recall}{Precision + Recall}$$

- Valuable for assessing model performance, especially on datasets with uneven class distributions.

Regression and Classification

ROC (Receiver Operating Characteristic) Curves

- ROC Curve plots True Positive Rate (TPR) against False Positive Rate (FPR).
- Helps in evaluating the performance of classification models at different thresholds.
- Area Under the Curve (AUC) measures the overall performance.
- Higher AUC indicates a better-performing model.
- Useful for comparing multiple models

Regression and Classification

ROC (Receiver Operating Characteristic) Curves

Ranking instead of classifying

Classifiers such as logistic regression can output a probability of belonging to a class (or something similar)

- We can use this to **rank** the different instances and take actions on the cases at top of the list
- We may have a **budget**, so we have to target most promising individuals
- Ranking enables to use different techniques for **visualizing** model performance

Regression and Classification

ROC (Receiver Operating Characteristic) Curves

ROC curves are a very general way to **represent and compare** the performance of different models
(on a binary classification task)

Observations

- (0,0): classify always negative
- (1,1) : classify always positive
- Diagonal line: random classifier
- Below diagonal line: worse than random classifier
- Different classifiers can be compared

Area Under the Curve (AUC): probability that a randomly chosen positive instance will be ranked ahead of randomly chosen negative instance

Regression and Classification

Multi Class Classification with Logistic Regression

- Logistic Regression can be extended to multi-class classification using techniques like One-vs-Rest (OvR) and Softmax Regression.
- One-vs-Rest: Trains a separate binary classifier for each class.
- Softmax Regression: Generalizes logistic regression to handle multiple classes.
- Outputs probability distribution over multiple classes.
- Suitable for tasks like image classification, text classification.

?

Thank You !!!