```
import pandas as pd

# Load the dataset
df = pd.________("students.csv")   # Fill in the function

# Display top 5 rows
print(df._____())               # Show first 5 rows

# Check shape of the data
print("Shape of Data:", df._______)  # Show number of rows and columns

# Check for missing values
print(df.__________())            # Function to check missing values

# Fill missing Age with mean
df['Age'] = df['Age'].fillna(df['Age'].__________())  # Fill with mean

# Remove duplicate records
df = df.________________()         # Drop duplicates

# Create a new column "Grade" based on Score
df['Grade'] = df['Score'].apply(lambda x: 'A' if x >= 80 else ('B' if x >= 60 else 'C'))

# Sort dataframe by Score in descending order
df_sorted = df.sort_values(by='_______', ascending=False)

# Filter students who passed
passed_students = df[df['Passed'] == _____]     # True or False?

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

```python
# Countplot of Gender
sns.___________(x='Gender', data=df)
plt.title("Gender Distribution")
plt.show()

# Boxplot of Score by Gender
sns.boxplot(x='Gender', y='_______', data=df)
plt.title("Score by Gender")
plt.show()

# Histogram of Scores
sns.histplot(df['Score'], bins=10, kde=True)
plt.title("Score Distribution")
plt.show()
```

---

sample super store data

```python
import pandas as pd

# Load data
df = pd._________("SampleSuperstore.csv")  # Load the file

# View top rows
print(df._____())  # Preview the data

# Basic info
df._________()  # View column types and missing data

# Convert 'Order Date' to datetime
df['Order Date'] = pd.to_datetime(df['____________'])

# Drop unnecessary columns (e.g., 'Postal Code' if present)
```

```
df = df.drop(['_______________'], axis=1)

# Check for duplicates
print("Duplicates:", df.___________())  # Count duplicates

# Calculate total sales per state
state_sales = df.groupby('_______')['Sales'].sum().sort_values(ascending=False)

# Find top 5 profitable sub-categories
top_profit_subcats =
df.groupby('_____________')['Profit'].sum().sort_values(ascending=False).head()

# Create a new column for Profit Margin (Profit/Sales)
df['Profit Margin'] = df['Profit'] / df['Sales']

import matplotlib.pyplot as plt
import seaborn as sns

# Sales distribution
sns.histplot(df['Sales'], bins=30, kde=True)
plt.title("Sales Distribution")
plt.show()

# Profit vs Discount scatterplot
sns.scatterplot(x='Discount', y='Profit', data=df)
plt.title("Profit vs Discount")
plt.show()

# Region-wise total sales
region_sales = df.groupby('Region')['Sales'].sum().reset_index()
sns.barplot(x='Region', y='Sales', data=region_sales)
plt.title("Sales by Region")
plt.show()
```

- Which **state** generates the highest sales?
- Which **sub-category** is the most profitable?
- Does **higher discount** always lead to **higher profit**?
- What kind of relationship do you observe between **quantity sold** and **profit**?
- Plot a chart showing **sales by category**.

Project: Predicting Customer Churn for a Telecom Company

Dataset: `Telco-Customer-Churn.csv`
This project includes:

- Data loading
- Data cleaning
- Exploratory Data Analysis (EDA)
- Feature engineering
- Model training
- Evaluation
- Interpretation

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


# Load dataset

df = pd.read_csv("Telco-Customer-Churn.csv")

df.head()

# Check data types and null values

df.info()


# Convert TotalCharges to numeric (some might be empty strings)

df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

```
# Drop missing values
df.dropna(inplace=True)

# Drop 'customerID' as it's not useful
df.drop('customerID', axis=1, inplace=True)

# Count plot for churn
sns.countplot(x='Churn', data=df)
plt.title("Churn Distribution")
plt.show()

# Boxplot of Monthly Charges vs Churn
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.title("Monthly Charges vs Churn")
plt.show()

# Correlation heatmap
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
# Convert categorical variables to dummy variables
df_encoded = pd.get_dummies(df, drop_first=True)

# Define features and target
X = df_encoded.drop('Churn_Yes', axis=1)
y = df_encoded['Churn_Yes']
from sklearn.model_selection import train_test_split

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

from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Build the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Confusion Matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Choosing the Right Feature Selection Method

- **Correlation Matrix**: Good for removing correlated features (e.g., Pearson correlation > 0.9).
- **Univariate Selection**: Use statistical tests for categorical data (like Chi-squared) or numerical data (like ANOVA).
- **RFE**: Great when combined with a model (like Random Forest or SVM) to recursively remove less important features.
- **Model-Based Feature Importance**: Tree-based models like Random Forest are great for identifying feature importance.
- **Lasso Regression**: Use when you want to shrink less important features to zero.
- **PCA**: Ideal when you want to reduce the dimensionality and keep the most important components.
- **Mutual Information**: Best when you suspect non-linear relationships between features and the target.

1. Correlation Matrix (for numerical data)

- **Purpose**: Identify highly correlated features and remove redundant ones.
- **Method**: Calculate the correlation between each pair of features and drop features that are highly correlated.

```
import seaborn as sns

import matplotlib.pyplot as plt


# Calculate correlation matrix

corr_matrix = df.corr()


# Display the heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title("Correlation Matrix")

plt.show()


# Drop features that are highly correlated (e.g., correlation > 0.9)

high_corr_features = set()

for i in range(len(corr_matrix.columns)):

    for j in range(i):

        if abs(corr_matrix.iloc[i, j]) > 0.9:  # Threshold can be adjusted

            colname = corr_matrix.columns[i]

            high_corr_features.add(colname)


df_reduced = df.drop(columns=high_corr_features)
```

2. Univariate Feature Selection

- **Purpose**: Use statistical tests to select the features that have the strongest relationship with the target variable.
- **Method**: Use methods like chi-squared test, ANOVA F-test, or mutual information for selection.

```
from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import chi2


# Select the top 10 features based on chi-squared test

X = df.drop('target', axis=1)  # Replace 'target' with your actual target column name

y = df['target']


selector = SelectKBest(chi2, k=10)

X_new = selector.fit_transform(X, y)


# Get the selected feature names

selected_features = X.columns[selector.get_support()]

print("Selected Features:", selected_features)
```

3. Recursive Feature Elimination (RFE)

- **Purpose**: Recursively removes features and builds a model on the remaining features to identify the most important ones.
- **Method**: It evaluates the importance of each feature based on a model (e.g., linear regression, decision trees) and eliminates the least important features.

```
from sklearn.feature_selection import RFE

from sklearn.ensemble import RandomForestClassifier


# Create a random forest classifier model

model = RandomForestClassifier()


# Create the RFE model and select top 10 features

rfe = RFE(model, 10)

X_rfe = rfe.fit_transform(X, y)


# Get the selected features

selected_features = X.columns[rfe.support_]

print("Selected Features:", selected_features)
```

4. Feature Importance Using Models

- **Purpose**: Use tree-based models like Random Forest, XGBoost, or Decision Trees, which provide feature importance scores.
- **Method**: Train a model and extract the feature importances.

```
from sklearn.ensemble import RandomForestClassifier


# Train the RandomForest model
model = RandomForestClassifier()
model.fit(X, y)


# Get the feature importance
importances = model.feature_importances_


# Sort the feature importances
sorted_idx = importances.argsort()


# Plot the top 10 features based on importance
plt.figure(figsize=(10, 8))
plt.barh(X.columns[sorted_idx][:10], importances[sorted_idx][:10])
plt.title('Top 10 Important Features')
plt.show()
```

L1 Regularization (Lasso Regression)

- **Purpose**: Lasso (L1 regularization) can be used to shrink the coefficients of less important features to zero, effectively performing feature selection.
- **Method**: Apply Lasso regression to find and eliminate features with zero or near-zero coefficients.

```
from sklearn.linear_model import Lasso


# Lasso regression model
lasso = Lasso(alpha=0.01)  # You can tune the alpha parameter
lasso.fit(X, y)


# Get the features selected (non-zero coefficients)
selected_features = X.columns[lasso.coef_ != 0]
print("Selected Features:", selected_features)
```

6. Principal Component Analysis (PCA) (for dimensionality reduction)

- **Purpose**: PCA is used to reduce the number of features by transforming the original features into a smaller set of components that explain most of the variance in the data.
- **Method**: Apply PCA and select the components that explain the most variance.

```
from sklearn.decomposition import PCA


# Apply PCA
pca = PCA(n_components=5)  # You can choose the number of components based on explained variance
X_pca = pca.fit_transform(X)


# Plot the explained variance ratio
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_)
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance by Principal Components')
plt.show()
```

7. Mutual Information

- **Purpose**: Measures the mutual dependence between two variables. It is useful for identifying non-linear relationships.
- **Method**: Use `mutual_info_classif` or `mutual_info_regression` from `sklearn.feature_selection`.

```
from sklearn.feature_selection import mutual_info_classif


# Calculate mutual information
mi = mutual_info_classif(X, y)


# Create a dataframe for mutual information values
mi_df = pd.DataFrame({'Feature': X.columns, 'Mutual Information': mi})
mi_df = mi_df.sort_values(by='Mutual Information', ascending=False)
print(mi_df)
```

---

8. Wrapper Methods

- **Purpose**: Use a machine learning algorithm to evaluate subsets of features and find the best subset.
- **Method**: Techniques like **Forward Selection**, **Backward Elimination**, and **Exhaustive Search**.

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LogisticRegression


# Logistic Regression as the model
model = LogisticRegression()


# Forward Selection
selector = SequentialFeatureSelector(model, n_features_to_select=10, direction='forward')
```

```
selector.fit(X, y)

# Selected features
selected_features = X.columns[selector.get_support()]
print("Selected Features:", selected_features)
```

1. Customer Churn Prediction

- **Description**: Predict whether a customer will churn or stay with a service based on various features (e.g., subscription details, usage patterns, demographics).
- **Skills**: Data preprocessing, classification, feature selection, model evaluation (e.g., Random Forest, Logistic Regression).
- **Dataset**: Telecom Churn Dataset, Bank Churn Dataset.

2. House Price Prediction

- **Description**: Predict the price of a house based on its features such as square footage, number of rooms, location, etc.
- **Skills**: Regression analysis, feature engineering, model evaluation (e.g., Linear Regression, XGBoost).
- **Dataset**: Boston Housing Dataset, Kaggle's House Prices dataset.

3. Sentiment Analysis on Social Media Data

- **Description**: Analyze tweets, reviews, or other social media data to classify sentiments (positive, negative, neutral).
- **Skills**: Natural Language Processing (NLP), text preprocessing, sentiment analysis, machine learning.
- **Dataset**: Twitter API, Yelp reviews, Amazon reviews.

4. Credit Card Fraud Detection

- **Description**: Identify fraudulent credit card transactions based on transaction data.
- **Skills**: Anomaly detection, classification models, imbalanced data handling.
- **Dataset**: Kaggle's Credit Card Fraud Detection Dataset.

5. Image Classification Using Deep Learning

- **Description**: Build a model to classify images into categories (e.g., dog vs. cat, facial recognition).
- **Skills**: Convolutional Neural Networks (CNNs), image preprocessing, model evaluation (e.g., accuracy, precision).
- **Dataset**: CIFAR-10, MNIST, ImageNet.

6. Recommendation System for E-commerce

- **Description**: Build a recommendation engine to suggest products to users based on their browsing/purchase history.
- **Skills**: Collaborative filtering, content-based filtering, matrix factorization.
- **Dataset**: Amazon Product Data, MovieLens.

7. Customer Segmentation Using Clustering

- **Description**: Segment customers into different groups based on features like purchasing behavior, demographics, etc.

- **Skills**: K-means clustering, PCA, unsupervised learning.
- **Dataset**: Mall Customer Segmentation Dataset.

8. Stock Price Prediction

- **Description**: Predict the future stock price of a company using historical data and technical indicators.
- **Skills**: Time series analysis, LSTM (Long Short-Term Memory networks), regression models.
- **Dataset**: Yahoo Finance, Quandl.

9. Traffic Prediction Using Weather and Historical Data

- **Description**: Predict traffic congestion or accidents based on weather conditions, historical traffic data, and special events.
- **Skills**: Time series analysis, feature engineering, regression models.
- **Dataset**: Traffic dataset, weather API data.

10. Sales Forecasting for a Retail Company

- **Description**: Predict future sales for a retail company based on historical sales data, promotions, holidays, etc.
- **Skills**: Time series forecasting, ARIMA, Prophet, XGBoost.
- **Dataset**: Kaggle’s Retail Sales Forecasting Dataset.

11. Fake News Detection

- **Description**: Classify news articles as "fake" or "real" using text classification techniques.
- **Skills**: NLP, text classification, feature extraction, machine learning models (e.g., SVM, Random Forest).
- **Dataset**: Fake News Dataset, Kaggle’s Fake News Dataset.

12. Human Activity Recognition Using Smartphone Data

- **Description**: Recognize human activities (walking, running, sitting) from smartphone sensor data.
- **Skills**: Time series data, feature engineering, classification (e.g., Random Forest, SVM).
- **Dataset**: UCI HAR dataset.

13. Anomaly Detection in Network Traffic

- **Description**: Detect unusual patterns or anomalies in network traffic (e.g., DDoS attacks).
- **Skills**: Anomaly detection, unsupervised learning, clustering, classification.
- **Dataset**: UNSW-NB15 Network Traffic Dataset.

14. Customer Lifetime Value Prediction

- **Description**: Predict the future revenue that a customer will generate during their relationship with a company.
- **Skills**: Regression models, time series forecasting, customer segmentation.
- **Dataset**: Retail transaction data, telecom data.

15. Healthcare Predictive Analytics

- **Description**: Predict the likelihood of a patient developing a particular disease (e.g., diabetes, heart disease) based on medical records and lifestyle factors.
- **Skills**: Classification models, feature selection, imbalanced class handling.
- **Dataset**: UCI Heart Disease dataset, Diabetes dataset.

16. Text Summarization Using NLP

- **Description**: Build a model that summarizes long articles into short, concise summaries.
- **Skills**: NLP, text summarization, sequence-to-sequence models.
- **Dataset**: News articles, Wikipedia.

17. Time Series Forecasting for Electricity Demand

- **Description**: Predict future electricity demand based on historical demand and external factors like temperature.
- **Skills**: Time series forecasting, ARIMA, LSTM.
- **Dataset**: UCI Electricity Consumption Dataset, Kaggle's Electricity Demand dataset.

18. Air Quality Prediction

- **Description**: Predict air quality index (AQI) based on weather data and pollutants.
- **Skills**: Regression models, time series analysis, feature engineering.
- **Dataset**: UCI Air Quality dataset.

19. Real-Time Object Detection in Video Streams

- **Description**: Build a system that can detect and label objects (e.g., cars, people, etc.) in video streams in real time.
- **Skills**: Computer vision, real-time processing, object detection (e.g., YOLO, SSD).
- **Dataset**: COCO, Open Images Dataset.

20. Predicting Movie Ratings

- **Description**: Build a recommendation system that predicts ratings for movies a user may like based on their preferences.
- **Skills**: Collaborative filtering, matrix factorization, content-based filtering.
- **Dataset**: MovieLens dataset.