THE DEVELOPER ARENA

PART 2: MONTH 2 –DATA ANALYSIS AND STATISTICAL METHODS

WEEK 0 5

EDA (Exploratory Data Analysis) is the first and most important step when working with any dataset. It's like opening a mystery novel — you don't jump to conclusions; you explore, observe, and ask questions.

EDA can be defined as:
A diagnostic test for your data's health.
A way to understand structure, spot errors, and find patterns.
The bridge between raw data and meaningful insights.

Objectives of EDA:

- Understand what data you're working with: types, dimensions, categories.
- Identify missing data, outliers, or errors.
- Summarize the main characteristics of data using statistics and visuals.
- Spot relationships between variables.
- Lay the groundwork for deeper analysis or machine learning.

Core Components of EDA:

1. Descriptive Statistics:

- These are numerical measures that summarize the main features of a dataset:
- Mean – the average value.
- Median – the middle value (helps deal with outliers).
- Mode – the most frequent value.
- Standard Deviation – how spread out the values are.
- Minimum & Maximum – boundaries of the data.
- Percentiles/Quartiles – to understand data distribution.
- Descriptive stats give you a quick glance at the data's shape and spread.

2. Data Types & Structure:

- Numerical (continuous or discrete)
- Categorical (nominal or ordinal)
- Textual or Temporal (like timestamps or freeform text)
- This helps you decide what type of analysis or visualization to apply.

3. Missing Values & Duplicates:

- Real-world data is messy. Common issues:
- Null or NaN values that can distort analysis.
- Duplicate entries which need to be removed or consolidated.
- Understanding the missing data mechanism (Missing Completely at Random, Missing at Random, or Not at Random) is key for proper handling.

4. Outliers

- Outliers are data points that deviate significantly from other observations.
- Could indicate errors, rare events, or important anomalies.
- Need to be handled carefully — removing outliers can sometimes erase valuable signals.

5. Correlation & Relationships:

- This step checks how variables are related to each other:
- Positive/Negative/No correlation.
- Helps in selecting features for modeling.
- Visualized using heatmaps or scatter plots.

6. Visual Exploration:

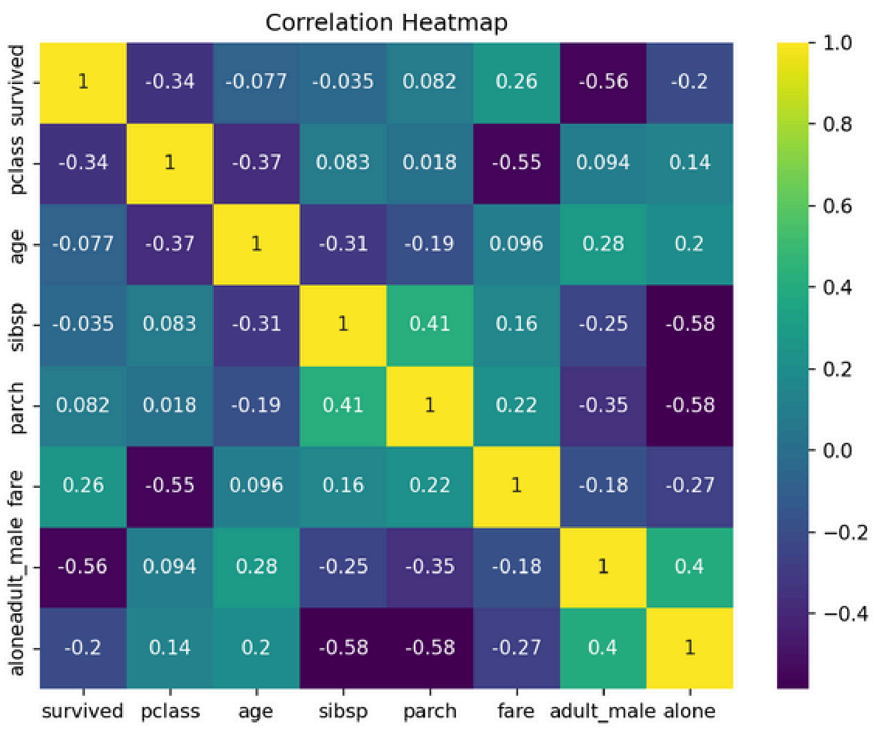
- Histograms → Distribution of values.
- Boxplots → Outliers, median, IQR.
- Scatterplots → Relationships between pairs of variables.
- Bar charts → Categorical comparisons.
- Line plots → Time series trends.
- Correlation matrix/heatmaps → Inter-variable relationships.

- **Hands-On Programs Developed:**

  Hands-On: Perform basic EDA on a dataset (e.g., summary statistics, correlation analysis).

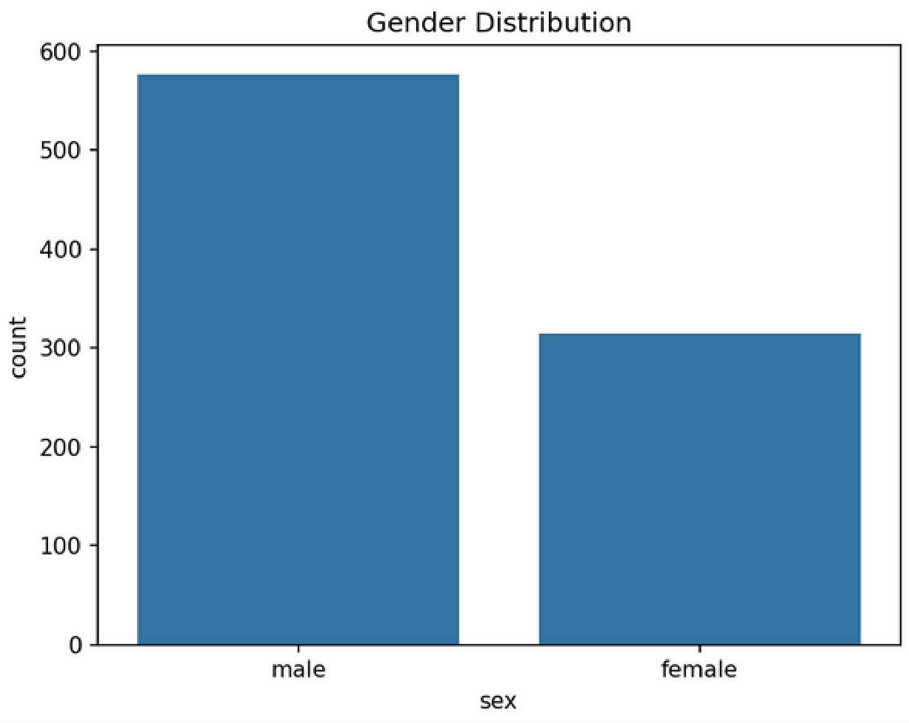
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
df = sns.load_dataset('titanic')

# Dataset shape & info
print("Shape:", df.shape)
print("\nInfo:")
print(df.info())

# Summary statistics
print("\nSummary Statistics:")
print(df.describe(include='all'))

# Missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Correlation analysis (numeric only)
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cm
plt.title("Correlation Heatmap")
plt.show()

# Visualizations
sns.countplot(x='sex', data=df)
plt.title("Gender Distribution")
plt.show()

sns.histplot(df['age'].dropna(), kde=True)
plt.title("Age Distribution")
plt.show()

sns.boxplot(x='pclass', y='age', data=df)
plt.title("Passenger Class vs Age")
plt.show()
```

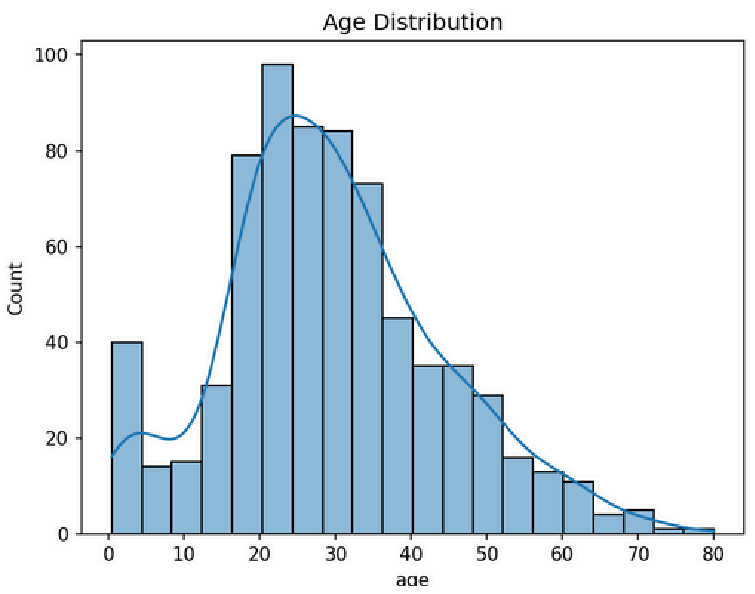
```
sns.countplot(x='survived', hue='sex', data=df)
plt.title("Survival by Gender")
plt.show()
```

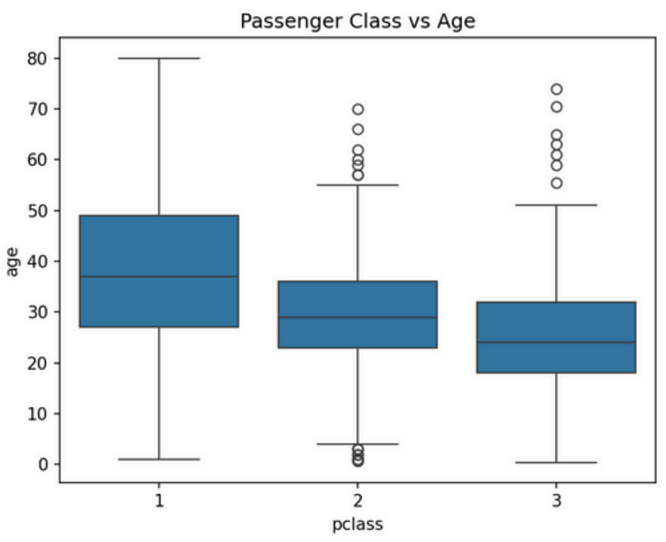


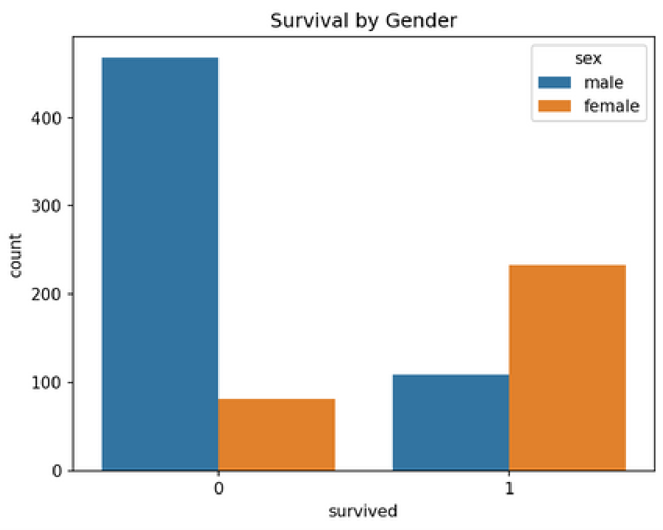
	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1	-0.34	-0.077	-0.035	0.082	0.26	-0.56	-0.2
pclass	-0.34	1	-0.37	0.083	0.018	-0.55	0.094	0.14
age	-0.077	-0.37	1	-0.31	-0.19	0.096	0.28	0.2
sibsp	-0.035	0.083	-0.31	1	0.41	0.16	-0.25	-0.58
parch	0.082	0.018	-0.19	0.41	1	0.22	-0.35	-0.58
fare	0.26	-0.55	0.096	0.16	0.22	1	-0.18	-0.27
adult_male	-0.56	0.094	0.28	-0.25	-0.35	-0.18	1	0.4
alone	-0.2	0.14	0.2	-0.58	-0.58	-0.27	0.4	1

- **Client Project:**




Age Distribution
Count
age
100
80
60
40
20
0
0
10
20
30
40
50
60
70
80


Passenger Class vs Age
age
pclass
80
70
60
50
40
30
20
10
0
1
2
3


Survival by Gender
count
survived
sex
male
female
400
300
200
100
0
0
1

- **Client Project:**

```
# EDA for Client Project — Student Performance
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("StudentsPerformance.csv")  # Make sure this file is in the same folder

# Basic structure
print(df.head())
print("\nInfo:")
print(df.info())
print("\nSummary:")
print(df.describe(include='all'))

# Check missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Rename columns (optional cleanup)
df.columns = [col.strip().lower().replace(" ", "_") for col in df.columns]

# Plot: Average scores by gender
df['average_score'] = df[['math_score', 'reading_score', 'writing_score']].mean(axis=1)
sns.boxplot(x='gender', y='average_score', data=df)
plt.title("Average Scores by Gender")
plt.show()

# Plot: Test scores vs parental education
sns.barplot(x='parental_level_of_education', y='average_score', data=df)
plt.xticks(rotation=45)
plt.title("Scores vs Parental Education")
plt.show()

# Correlation Heatmap
numeric_cols = ['math_score', 'reading_score', 'writing_score']
sns.heatmap(df[numeric_cols].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation between Exam Scores")
plt.show()
```

SUMMARY OF WEEK 05

EDA:

Exploratory Data Analysis (EDA) is the process of examining and understanding a dataset before applying models. It helps you see the shape, structure, patterns, and quirks hidden in raw data.

Techniques for Summarizing Datasets

1. Data Overview:

- Shape: Rows × Columns
- Data Types: Categorical, numerical, boolean, datetime
- Missing Values: Check and handle them

2. Descriptive Statistics
For numerical columns:

- Mean: Average value
- Median: Middle value (robust to outliers)
- Mode: Most frequent value
- Standard Deviation: Spread/variability
- Min / Max / Range: Distribution limits
- Percentiles: e.g., 25th, 50th, 75th (IQR = Q3 - Q1)

3. Categorical Summary
Count of each category using frequency tables or bar charts
Uniqueness and class distribution

4. Common EDA Tasks

- Data Cleaning: Handle missing values, fix types, remove duplicates
- Univariate Analysis: Analyze one feature at a time
- Bivariate Analysis: Relationships between two features
- Multivariate Analysis: Trends across multiple features
- Outlier Detection: Use boxplots, Z-scores, IQR method
- Distribution Analysis: Histograms, KDE plots
- Correlation Analysis: Pearson/Spearman + heatmaps

5. EDA Visual Tools

- Histograms – Distribution of numerical variables
- Boxplots – Spread and outliers
- Bar Charts – Category counts
- Scatterplots – Relationships between two variables

WEEK 0 6

Probability & Statistical Testing

Distributions describe how values in a dataset are spread — they help us predict, simulate, and compare real-world scenarios.

Normal Distribution (Gaussian):

Bell-shaped, symmetric
Defined by:
Mean (μ): center
Standard deviation (σ): spread
About 68% of values lie within ±1σ, 95% within ±2σ

Real-World: Heights, IQ scores, test marks

Poisson Distribution:

Models count of events in fixed intervals
Used when events are independent and rare

Real-World: Calls per minute, defects per unit, email arrivals

2. Hypothesis Testing

Concept	Meaning
Null Hypothesis (H_0)	Default claim (e.g., no difference)
Alternative (H_1)	What you aim to prove
p-value	Probability of seeing the result
α (alpha)	Significance level, usually 0.05
Reject H_0 if	$p < \alpha$

T-Test
Compare means of groups:
Types:
One-Sample T-Test → sample vs population mean
Independent T-Test → 2 different groups
Paired T-Test → same group, before vs after

Chi-Square Test
For categorical data
Tests if distributions are significantly different

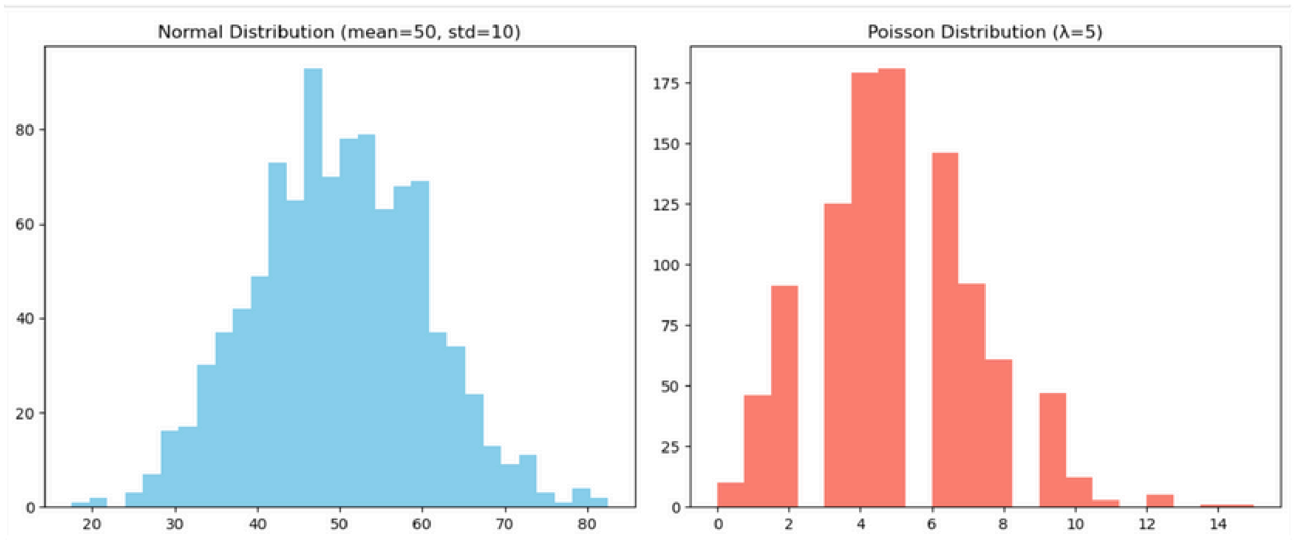
3. Confidence Intervals (CI):

A range that's likely to contain the true population parameter.
E.g., "We're 95% confident the avg. time spent is between 4.5 and 5.1 hours."
Formula (simplified):
CI = sample mean ± (critical value × standard error)

Test	Used When	Variable Type
t-test	Comparing two means	Continuous
z-test	Known population std dev	Continuous
Chi-squared	Comparing frequencies	Categorical
ANOVA	Comparing >2 means	Continuous
Mann-Whitney U	Non-parametric t-test	Ordinal/Continuous
Paired t-test	Same group, before/after	Continuous

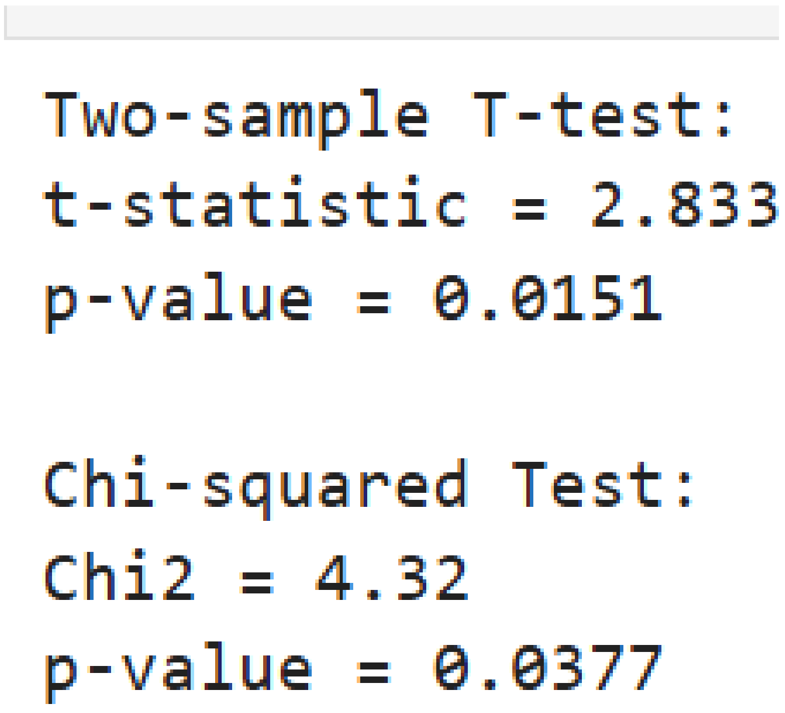
HANDS-ON PROJECT: Simulate & Analyze

```
import numpy as np
import matplotlib.pyplot as plt

# Simulate normal distribution data (e.g., product ratings)
normal_data = np.random.normal(loc=50, scale=10, size=1000)

# Simulate Poisson distribution data (e.g., customer service calls per hour)
poisson_data = np.random.poisson(lam=5, size=1000)

# Plot histograms
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.hist(normal_data, bins=30, color='skyblue')
plt.title("Normal Distribution (mean=50, std=10)")

plt.subplot(1, 2, 2)
plt.hist(poisson_data, bins=20, color='salmon')
plt.title("Poisson Distribution (λ=5)")

plt.tight_layout()
plt.show()
```



Hypothesis Testing: t-test and Chi-squared

```
from scipy import stats

# Two groups: customer satisfaction scores from two regions
group_a = [60, 65, 68, 70, 72, 75, 78]
group_b = [55, 58, 60, 62, 64, 66, 67]

# Perform two-sample t-test
t_stat, p_val = stats.ttest_ind(group_a, group_b)
print("Two-sample T-test:")
print("t-statistic =", round(t_stat, 3))
print("p-value =", round(p_val, 4))

# Chi-squared test: Ad clicked (Yes/No) for 2 campaigns
#       Clicked, Not Clicked
observed = [[30, 10],    # Campaign A
            [20, 20]]    # Campaign B

chi2, p, dof, expected = stats.chi2_contingency(observed)
print("\nChi-squared Test:")
print("Chi2 =", round(chi2, 2))
print("p-value =", round(p, 4))
```



```
Two-sample T-test:
t-statistic = 2.833
p-value = 0.0151

Chi-squared Test:
Chi2 = 4.32
p-value = 0.0377
```

Client Project -Business Strategy Comparison

```
import pandas as pd

# Simulated revenue per day for each strategy
data = {
    'Strategy': ['A'] * 7 + ['B'] * 7,
    'Revenue': [210, 220, 230, 250, 245, 225, 240,   # Strategy A
                190, 200, 195, 205, 198, 202, 210]   # Strategy B
}

df = pd.DataFrame(data)
print(df)
# Separate groups
group_a = df[df['Strategy'] == 'A']['Revenue']
group_b = df[df['Strategy'] == 'B']['Revenue']

# Perform t-test
t_stat, p_val = stats.ttest_ind(group_a, group_b)
print("\nBusiness Strategy T-Test:")
print("Strategy A Mean Revenue:", round(group_a.mean(), 2))
print("Strategy B Mean Revenue:", round(group_b.mean(), 2))
print("t-statistic:", round(t_stat, 3))
print("p-value:", round(p_val, 4))

# Conclusion
if p_val < 0.05:
    print("✅ Significant difference: Strategy A is better.")
else:
    print("❌ No significant difference between strategies.")
```

SUMMARY OF WEEK 06

Probability Distributions

Normal Distribution: Continuous, symmetric, bell-shaped curve; used for natural measurements.
Poisson Distribution: Discrete; used to model number of events in a fixed interval.

Hypothesis Testing

Process of testing assumptions about a population using sample data.
Involves null hypothesis (H_0) and alternative hypothesis (H_1).
Based on p-value and significance level (usually 0.05).
Common tests: t-test (for means), chi-squared test (for categorical data).

Confidence Intervals

A range of values that likely contains the population parameter.
Example: 95% CI means we are 95% confident the true value lies within the interval.

Hands-On Practice

Simulated normal and Poisson distributions using NumPy.
Visualized the distributions using histograms.
Performed hypothesis testing (t-test and chi-squared) using SciPy on sample data.

Client Project

Compared two business strategies (A & B) using revenue data.
Conducted a t-test to analyze differences in average performance.
Conclusion: Strategy A outperformed Strategy B (p-value < 0.05 indicated significant difference).

WEEK 0 7

Introduction to Machine Learning (ML)

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that enables systems to learn from data and improve their performance on a task without being explicitly programmed. It focuses on developing algorithms that can identify patterns in data, make decisions, or make predictions based on learned information.

Types of Machine Learning

a. Supervised Learning
In supervised learning, the algorithm is trained on a labeled dataset, meaning that each training example is paired with a corresponding output label. The goal is to learn a mapping from inputs to outputs so the model can predict outputs for new, unseen data.
Examples:
Predicting house prices (regression)
Classifying emails as spam or not spam (classification)

b. Unsupervised Learning
Unsupervised learning involves training algorithms on data without labeled outputs. The goal is to explore the structure of the data, find patterns, or group similar data points.
Examples:
Customer segmentation
Market basket analysis
Anomaly detection

c. Semi-Supervised Learning
This approach combines a small amount of labeled data with a large amount of unlabeled data. It is useful when labeling data is expensive or time-consuming.

d. Reinforcement Learning
In reinforcement learning, an agent learns to make decisions by interacting with an environment. The agent receives rewards or penalties based on its actions and aims to maximize cumulative reward.

Types of ML Tasks and Algorithms

Task Type	Description	Example Algorithms
Regression	Predict a continuous value	Linear Regression, Ridge, Lasso
Classification	Predict a categorical label	Logistic Regression, Decision Tree,
Clustering	Group similar data points	K-Means, DBSCAN,

Scikit-learn

Scikit-learn (sklearn) is a widely used open-source machine learning library in Python. It provides a range of tools for:

- Data preprocessing
- Model selection and evaluation
- Implementing various ML algorithms (regression, classification, clustering)

Scikit-learn ML Workflow:

- Import and load the dataset
- Preprocess the data (handle missing values, scale, encode, etc.)
- Split the dataset into training and testing sets
- Choose and initialize the ML model
- Train the model on the training set
- Make predictions on the test set
- Evaluate the model using appropriate metrics

HANDS-ON PROJECT: hands_on_salary_data

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Load the dataset
df = pd.read_csv("hands_on_salary_data.csv")
print("Dataset:")
print(df.head())

# Step 2: Prepare features and target
X = df[['Age']]  # Features (must be 2D)
y = df['Salary'] # Target

# Step 3: Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Make predictions
y_pred = model.predict(X_test)

# Step 6: Evaluate the model
print(f"R2 Score: {r2_score(y_test, y_pred):.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}")

# Step 7: Visualize
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Regression Line')
plt.title('Linear Regression - Salary vs Age')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

CLIENT PROJECT –client_house_price_data

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error
import matplotlib.pyplot as plt

# Step 1: Load the client dataset
df = pd.read_csv("client_house_price_data.csv")
print("Sample Data:\n", df.head())

# Step 2: Define features and target
X = df[['Area', 'Bedrooms', 'Bathrooms', 'Age']]
y = df['Price']

# Step 3: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Predict and evaluate
y_pred = model.predict(X_test)

print("\nModel Evaluation:")
print(f"R2 Score: {r2_score(y_test, y_pred):.2f}")
print(f"Mean Absolute Error: {mean_absolute_error(y_test, y_pred):,.2f}")

# Step 6: Visualize actual vs predicted prices
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, color='green')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', linewidth=2)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted House Prices")
plt.grid(True)
plt.tight_layout()
plt.show()
```

SUMMARY OF WEEK 07

In Week 7, we began our journey into the field of Machine Learning (ML). The theory sessions focused on understanding what machine learning is and how it enables computers to learn from data without being explicitly programmed. We explored the major types of ML: supervised learning and unsupervised learning.
In supervised learning, the model is trained on a dataset that contains input-output pairs (i.e., labeled data). The model learns the relationship between the features (inputs) and the target (output) so it can make predictions on new, unseen data. Common examples include predicting house prices, student scores, or whether an email is spam or not. In contrast, unsupervised learning deals with data that has no labels. The model tries to find hidden patterns or groupings in the data. Clustering (e.g., customer segmentation) is a popular technique under unsupervised learning.

We also learned about the main categories of ML algorithms:
Regression – used when the target variable is continuous (e.g., salary prediction)
Classification – used when the target is categorical (e.g., predicting whether a customer will buy or not)
Clustering – used to group similar data without using labels (e.g., grouping customers based on behavior)
As part of the theory, we were introduced to scikit-learn (sklearn), a powerful Python library used for building and evaluating ML models. We discussed its key modules and functions, such as train_test_split for splitting data, LinearRegression for regression tasks, and metrics like mean squared error and R^2 score for evaluating performance.

Hands-On Activity

- Loaded a small dataset with Age and Salary columns.
- Visualized the data using scatter plots to understand the relationship.
- Split the data into training and testing sets using train_test_split.
- Trained a LinearRegression model.
- Evaluated the model's predictions using R^2 score and plotted the regression line.
- Through this exercise, we learned how to prepare data, train a model, test it, and interpret its performance.

WEEK 0 8

Model Evaluation and Tuning

1. Performance Metrics

Once a model is built, it is important to evaluate how well it performs on unseen data. Several metrics are used for this purpose:

Accuracy: This measures the proportion of correctly predicted results out of all predictions. While simple and widely used, accuracy may not always give the full picture, especially when the dataset is imbalanced.

Precision: Precision tells us how many of the predicted positive cases were actually correct. It is particularly important when the cost of false positives is high (e.g., in spam detection).

Recall: Recall indicates how many of the actual positive cases were identified correctly. It becomes crucial in cases where missing a positive case is costly (e.g., in disease diagnosis).

F1 Score: This is the harmonic mean of precision and recall. It provides a balanced measure when we want to consider both false positives and false negatives. It is especially helpful in situations involving imbalanced classes.

2. Confusion Matrix

A confusion matrix is a simple but informative tool that helps us understand the types of errors our model is making. It displays true positives, true negatives, false positives, and false negatives in a tabular form. From this matrix, we can derive all the other metrics mentioned above.

- True Positives (TP): Correctly predicted positive class.
- True Negatives (TN): Correctly predicted negative class.
- False Positives (FP): Incorrectly predicted as positive.
- False Negatives (FN): Incorrectly predicted as negative.

3. Cross-Validation

Rather than evaluating the model based on a single train-test split, we use cross-validation to get a more robust estimate of model performance. The most commonly used method is k-fold cross-validation, where the dataset is divided into k equal parts. The model is trained on k–1 parts and validated on the remaining part. This process is repeated k times, and the results are averaged. This helps in reducing the risk of overfitting or underfitting and gives us a better understanding of the model's generalization ability.

4. Hyperparameter Tuning

Models often have hyperparameters that control their behavior but are not learned from the data. Examples include the depth of a decision tree or the number of neighbors in a KNN model. Choosing the right hyperparameters can significantly improve a model's performance. To automate this process, we use tools like GridSearchCV. GridSearchCV tries out multiple combinations of hyperparameters and evaluates each using cross-validation. The combination that performs the best is selected for the final model.

Default hyperparameter values may not work well for all datasets. Without proper tuning:

- A model may underfit (too simple, poor training accuracy)
- Or overfit (memorizes training data, poor generalization)
- Or train too slowly or inefficiently

Week 8: Hands-On
Evaluate Model Using Accuracy, Precision, Recall, F1-Score

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Step 1: Load dataset
df = pd.read_csv("hands_on_data.csv")
print("Sample Data:\n", df.head())

# Step 2: Define features and target
X = df.drop('Outcome', axis=1)
y = df['Outcome']

# Step 3: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train the model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Step 5: Make predictions
y_pred = model.predict(X_test)

# Step 6: Evaluate performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))
```

Week 8:Client Project

Tune a model to improve its performance.

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt

# Step 1: Load the client dataset
df = pd.read_csv("C:/Users/Radha/Downloads/client_house_price_data.csv")
print("Sample Data:\n", df.head())

# Step 2: Define features and target
X = df[['Area', 'Bedrooms', 'Bathrooms', 'Age']]
y = df['Price']

# Step 3: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Define model and parameters
ridge = Ridge()
params = {'alpha': [0.01, 0.1, 1, 10, 100]}

# Step 5: GridSearchCV
grid = GridSearchCV(ridge, params, cv=5, scoring='r2')
grid.fit(X_train, y_train)

# Step 6: Evaluate best model
best_model = grid.best_estimator_
y_pred = best_model.predict(X_test)

print("Best Alpha:", grid.best_params_['alpha'])
print("R2 Score:", r2_score(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
plt.scatter(y_test, y_pred, color='green')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted")
plt.grid()
plt.show()
```

SUMMARY OF WEEK 08

Objective:

To create a dashboard for visualizing relationships between features in a dataset using scatter plots, histograms, boxplots, and a correlation heatmap. This helps in understanding the structure and distribution of data during Exploratory Data Analysis (EDA).

Dataset Used:

Palmer Penguins Dataset (built-in in Seaborn) — contains measurements such as bill length & depth, flipper length, body mass, and species.

Visualizations Created:

Scatter Plot: Showed the relationship between bill length and bill depth. Each species had a distinct pattern.
Histogram: Displayed the distribution of flipper length. Most penguins had lengths around 200 mm.
Boxplot: Compared body mass across different species. Adelie penguins tended to be lighter.
Heatmap: Showed correlations between numeric features — flipper length and body mass were highly correlated.
Pairplot: Gave an overall snapshot of multiple feature relationships and species groupings.

What I Learned:

How to use Seaborn for attractive, statistical data visualizations
Importance of using EDA techniques before model building
How to detect patterns, trends, and correlations using graphs
How visualizations help in making data-driven decisions

Conclusion:

This dashboard offered a clear visual summary of the dataset, highlighting important relationships and distributions. Such visual tools are essential for data analysts and data scientists to draw meaningful insights and prepare data for machine learning models.