📘 Day 4 – Working with Machine Learning Models Using the California Housing Dataset

📅 **Date:** June 27, 2025

🔷 Key Takeaways

Today's session introduced us to the **core principles of Machine Learning**, along with hands-on experience of building a simple **regression model** using the **California Housing Dataset**. We explored how **Pandas**, **NumPy**, and **Scikit-learn** work together to help us prepare, train, and evaluate models efficiently.

🤖 What is Machine Learning?

Machine Learning (ML) is a technique under Artificial Intelligence that allows machines to learn from historical data and predict outcomes without hardcoding every rule. Instead of being explicitly programmed, these systems **adapt** by analyzing patterns in data.

🔍 Types of Machine Learning:

1. **Supervised Learning**
   - Works with labeled data
   - Used for tasks like **regression** (predicting numbers) and **classification** (categorizing outcomes)
   - Example: Predicting house prices based on income, population, etc.
2. **Unsupervised Learning**
   - Works without labeled output
   - Focuses on finding hidden patterns or clusters
   - Example: Grouping customers based on buying behavior
3. **Reinforcement Learning**
   - Learning through actions and receiving feedback in the form of rewards
   - Often used in gaming, robotics, and self-driving cars

🛠 Practical ML Project – California Housing Dataset

We used a built-in dataset provided by Scikit-learn called **California Housing**, which includes housing data like average number of rooms, population, median income, etc. The goal was to **predict median house value** using these features.

□ Step-by-Step Python Implementation

✅ 1. Importing Necessary Libraries

python

CopyEdit

```
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import joblib
```

✅ 2. Loading the Dataset

python

CopyEdit

```
california = fetch_california_housing()
X = california.data
y = california.target
```

✅ 3. Creating a DataFrame

python

CopyEdit

```
df = pd.DataFrame(X, columns=california.feature_names)
print(df.head())
```

This allows us to explore the features like MedInc, AveRooms, HouseAge, etc.

✅ 4. Splitting the Data for Training and Testing

python

CopyEdit

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

We reserved 20% of the data for testing so that we can validate our model after training.

✅ 5. Creating and Fitting the Model

python

CopyEdit

```
model = LinearRegression()
model.fit(X_train, y_train)
```

✅ 6. Making Predictions

python

CopyEdit

```
y_pred = model.predict(X_test)
```

✅ 7. Evaluating the Model

python

CopyEdit

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Model Coefficients:")
for name, coef in zip(california.feature_names, model.coef_):
    print(f"{name}: {coef:.2f}")

print(f"\nMean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")
```

Mean Squared Error (MSE): Measures how far off our predictions are on average
R² Score: Tells us how much of the variance in target values is explained by our model

✅ 8. Saving the Trained Model

python
CopyEdit

```
joblib.dump(model, 'housing_model.pkl')
print("Model saved as 'housing_model.pkl'")
```

We used joblib to export the trained model so it can be reused without retraining.

📊 Results

- The model calculated **individual coefficients** (weights) for each feature to understand their impact on housing prices.
- Using mean_squared_error and r2_score, we evaluated the model’s accuracy and reliability.
- Finally, the trained model was saved in .pkl format, making it easy to load and reuse in future predictions or applications.

💡 Skills Applied Today

- Loading and interpreting real-world datasets
- Data splitting and regression modeling
- Model evaluation using proper metrics
- Exporting models for reuse using joblib
- Working with Scikit-learn's machine learning pipeline