Lesson: Supervised Learning - Linear Regression and Other Models

In this session, we will go over **linear regression** and some other simple regression models to predict numerical values. We will also discuss how to evaluate these models using error metrics such as **MSE**, **MAE**, and **R-squared**.

Error Metrics: Simplified Explanation

1. Mean Absolute Error (MAE)

MAE measures the average of absolute errors between actual and predicted values.

- **Formula:**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Simplified Example:**
  - Actual values: [100, 200, 300].
  - Predicted values: [110, 190, 310].
  - Errors: |100-110|, |200-190|, |300-310| = [10, 10, 10].
  - MAE = (10 + 10 + 10) / 3 = **10**.

---

2. Mean Squared Error (MSE)

MSE is the average of the squared errors between actual and predicted values. Squaring emphasizes large errors.

- **Formula:**

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

- **Simplified Example:**
  - Actual values: [100, 200, 300].
  - Predicted values: [110, 190, 310].
  - Squared errors: (100-110)2^2, (200-190)2^2, (300-310)2^2 = [100, 100, 100].
  - MSE = (100 + 100 + 100) / 3 = **100**.

---

3. R-squared (Coefficient of Determination)

R-squared measures how well the regression model explains the variance in the data.

- **Values:** R-squared ranges from 0 to 1:
  - **0:** The model does not explain the variance.
  - **1:** The model perfectly explains the variance.
- **Formula:**

$$R^2 = 1 - \frac{\text{SS}_{\text{residual}}}{\text{SS}_{\text{total}}}$$

Now, let's implement these metrics in our models.

Part 1: 5 Simple Examples of Linear Regression

Example 1: Predicting Student Scores Based on Study Hours

Dataset:

Study Hours	Score
1	50
2	55
3	65
4	70
5	85

Code:

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

# Dataset
data = {'Study Hours': [1, 2, 3, 4, 5],
        'Score': [50, 55, 65, 70, 85]}
df = pd.DataFrame(data)

# Features and Target
X = df[['Study Hours']]
y = df['Score']
```

```
# Train Model
model = LinearRegression()
model.fit(X, y)

# Predictions
y_pred = model.predict(X)

# Evaluate Model
mae = mean_absolute_error(y, y_pred)
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

print("MAE:", mae)
print("MSE:", mse)
print("R-squared:", r2)

# Visualize
plt.scatter(X, y, color='blue', label='Actual')
plt.plot(X, y_pred, color='red', label='Prediction')
plt.title('Study Hours vs Score')
plt.xlabel('Study Hours')
plt.ylabel('Score')
plt.legend()
plt.show()
```

Example 2: Predicting House Prices Based on Area

Area (sq ft)	Price ($)
500	100000
1000	150000
1500	200000
2000	250000
2500	300000

Use the same approach as **Example 1**, replacing the dataset. This shows how **area** impacts house prices.

Example 3: Predicting Car Mileage Based on Engine Size

Engine Size (L)	Mileage (MPG)
1.0	40
1.5	35
2.0	30
2.5	25
3.0	20

Visualize how engine size negatively impacts mileage using linear regression.

Example 4: Predicting Employee Salaries Based on Experience

Experience (Years)	Salary ($)
1	40000
2	50000
3	60000
4	70000
5	80000

Train a linear regression model to show how salaries increase with experience.

Example 5: Predicting Rent Based on Apartment Size

Apartment Size (sq ft)	Rent ($)
600	1200
800	1600
1000	2000
1200	2400
1400	2800

Show the linear relationship between apartment size and rent.

Part 2: Other Regression Models

---

1. Decision Tree Regression

Decision Tree Regression splits the dataset into regions and fits predictions based on averages within those regions.

Example: Predicting House Prices Based on Area

```
from sklearn.tree import DecisionTreeRegressor

# Dataset
data = {'Area': [500, 1000, 1500, 2000, 2500],
        'Price': [100000, 150000, 200000, 250000, 300000]}
df = pd.DataFrame(data)

# Features and Target
```

```
X = df[['Area']]
y = df['Price']

# Train Model
tree_model = DecisionTreeRegressor()
tree_model.fit(X, y)

# Predictions
y_tree_pred = tree_model.predict(X)

# Evaluate
mae = mean_absolute_error(y, y_tree_pred)
mse = mean_squared_error(y, y_tree_pred)
r2 = r2_score(y, y_tree_pred)

print("Decision Tree - MAE:", mae)
print("Decision Tree - MSE:", mse)
print("Decision Tree - R-squared:", r2)
```

---

2. Random Forest Regression

Random Forest Regression builds multiple decision trees and averages their predictions.

```
from sklearn.ensemble import RandomForestRegressor

# Train Model
forest_model = RandomForestRegressor(n_estimators=10, random_state=42)
forest_model.fit(X, y)

# Predictions
y_forest_pred = forest_model.predict(X)

# Evaluate
mae = mean_absolute_error(y, y_forest_pred)
mse = mean_squared_error(y, y_forest_pred)
r2 = r2_score(y, y_forest_pred)

print("Random Forest - MAE:", mae)
print("Random Forest - MSE:", mse)
print("Random Forest - R-squared:", r2)
```

---

3. Support Vector Regression (SVR)

SVR fits a regression line within a margin of tolerance.

```
from sklearn.svm import SVR

# Train Model
svr_model = SVR(kernel='linear')
svr_model.fit(X, y)

# Predictions
y_svr_pred = svr_model.predict(X)
```

```
# Evaluate
mae = mean_absolute_error(y, y_svr_pred)
mse = mean_squared_error(y, y_svr_pred)
r2 = r2_score(y, y_svr_pred)

print("SVR - MAE:", mae)
print("SVR - MSE:", mse)
print("SVR - R-squared:", r2)
```

Summary

- **Linear Regression** works well for simple, linear relationships.
- **Decision Trees** and **Random Forests** handle non-linear relationships better.
- Use **MAE**, **MSE**, and **R-squared** to evaluate your models' accuracy and fit.
- Encourage students to try these models on different datasets to see their strengths and weaknesses!