TP : Machine learning with python (Prediction)

1 Introduction

This laboratory session focuses on developing a machine learning model to predict car prices. We'll work through a complete machine learning pipeline, from data preprocessing to model evaluation.

2 Step 1: Environment Setup and Data Creation

2.1 Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import warnings
warnings.filterwarnings('ignore')
```

2.2 Dataset Creation

We'll create a synthetic dataset that mimics real-world car sales data. Our features include:

- Brand (categorical): Toyota, Honda, Ford, BMW, Mercedes
- Age (numerical): 0-20 years
- Mileage (numerical): Normal distribution around 50,000
- Engine Size (numerical): 1.4L to 3.0L

```
# Generate sample data
n_samples = 1000
car_brands = ['Toyota', 'Honda', 'Ford', 'BMW', 'Mercedes']
age = np.random.randint(0, 20, n_samples)
mileage = np.random.normal(50000, 20000, n_samples)
engine_size = np.random.choice([1.4, 1.6, 1.8, 2.0, 2.4, 3.0],
                               n_samples)
brand = np.random.choice(car_brands, n_samples)
```

3 Step 2: Data Exploration

3.1 Basic Data Analysis

First, we examine our dataset's basic characteristics:

```
print(df.info())
print(df.describe())
print(df.isnull().sum())
```

Key aspects to analyze:

- Data types of each column
- Basic statistics (mean, std, min, max)
- Missing values
- Value distributions

3.2 Data Visualization

Create four key visualizations:

```
plt.figure(figsize=(15, 10))

# Price Distribution
plt.subplot(2, 2, 1)
sns.histplot(data=df, x='price')
plt.title('Distribution of Car Prices')

# Price by Brand
plt.subplot(2, 2, 2)
sns.boxplot(data=df, x='brand', y='price')
plt.title('Price Distribution by Brand')

# Price vs Age
plt.subplot(2, 2, 3)
sns.scatterplot(data=df, x='age', y='price')
plt.title('Price vs Age')

# Price vs Mileage
plt.subplot(2, 2, 4)
sns.scatterplot(data=df, x='mileage', y='price')
plt.title('Price vs Mileage')

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

4 Step 3: Data Preprocessing

4.1 Handling Missing Values

We handle missing values in the mileage column using mean imputation:

```
df['mileage'] = df['mileage'].fillna(df['mileage'].mean())
```

Alternative approaches include:

- Median imputation (for skewed distributions)
- Mode imputation (for categorical data)
- Interpolation (for time series data)
- backward and fillward imputation
- Removal of rows with missing values
- Give it a default value (in some cases)

4.2 One-Hot Encoding

Transform categorical 'brand' feature into numerical format:

```
df_encoded = pd.get_dummies(df, columns=['brand'],
                            prefix='brand')
```

Example transformation:

Original	Transformed			
brand	brand_BMW	brand_Ford	brand_Honda	brand_Toyota
BMW	1	0	0	0
Toyota	0	0	0	1

Table 1: One-Hot Encoding Example

5 Step 4: Model Development

5.1 Data Splitting

Split data into training (80%) and testing (20%) sets:

```
X = df_encoded.drop('price', axis=1)
y = df_encoded['price']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

5.2 Model Training

Train a linear regression model:

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

The linear regression model follows the formula:

$$Price = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon$$

Where:

- β_0 is the intercept
- β_i are the coefficients
- x_i are the feature values
- ϵ is the error term

6 Step 5: Model Evaluation

6.1 Performance Metrics

Calculate key performance metrics:

```
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

Metrics Formulas:

- R² Score:

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

- Mean Squared Error:

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

- Root Mean Squared Error:

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

6.2 Feature Importance Analysis

Analyze coefficient values to understand feature importance:

```
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.coef_
})
feature_importance = feature_importance.sort_values(
    'Coefficient', key=abs, ascending=False)
```

7 Step 6: Making Predictions

Test the model with a new sample:

```
new_sample = pd.DataFrame({
    'age': [5],
    'mileage': [45000],
    'engine_size': [2.0],
    'brand_BMW': [0],
    'brand_Ford': [0],
    'brand_Honda': [0],
    'brand_Mercedes': [0],
    'brand_Toyota': [1]
})

prediction = model.predict(new_sample)
```

8 Practice Exercise

You will work with the **5G-Energy Consumption** dataset provided by the International Telecommunication Union (ITU) in 2023. This dataset was part of a global challenge for data scientists to develop machine learning solutions for 5G energy consumption modeling. The dataset can be accessed at the following link: 5G-Energy Consumption Dataset.

Problem Statement

Network operational expenditure (OPEX) accounts for approximately 25% of total telecom operator costs, with 90% spent on energy bills. The radio access network (RAN), particularly base stations (BSs), consumes more than 70% of this energy.

Objective: Build and train a machine learning model to estimate energy consumption by different 5G base stations, considering:

- Various engineering configurations

- Traffic conditions
- Energy-saving methods

Tasks

1. Data Exploration and Preprocessing

1. Perform basic data exploration:
    - Display dataset information (shape, datatypes, basic statistics)
    - Create a pandas profiling report
    - Analyze missing values and data distribution
2. Data Cleaning:
    - Handle missing and corrupted values
    - Remove duplicates if present
    - Detect and handle outliers
    - Identify numerical and categorical features
3. Feature Engineering:
    - Apply one-hot encoding for categorical features

Example of One-Hot Encoding

Original Data:

```
| site_id | technology | power_state |
|---------|------------|-------------|
| A1      | 4G         | active      |
| A2      | 5G         | sleep       |
| A3      | 4G         | active      |
```

After One-Hot Encoding:

```
| site_id | technology_4G | technology_5G | power_state_active | power_state_sleep |
|---------|---------------|---------------|--------------------|-------------------|
| A1      | 1             | 0             | 1                  | 0                 |
| A2      | 0             | 1             | 0                  | 1                 |
| A3      | 1             | 0             | 1                  | 0                 |
```

2. Model Development

1. Feature Selection:
    - Choose appropriate features for your model
    - Select your target variable
2. Data Splitting:
    - Split the dataset into training and test sets
3. Model Implementation:
    - Implement linear regression

3. Model Evaluation

1. Performance Assessment:
    - Calculate and interpret the R2score and the MSE metrics on the test set
    - Analyze model strengths and weaknesses
2. Model Testing:
    - Test your model with a custom input row
    - Interpret the results

Note

Remember to:

- Document all your decisions
- Include comments in your code
- Handle errors appropriately