**Exercise: Feature Extraction and Selection in Python**

**Objective:**

* Understand how to perform feature extraction and selection from a dataset using Python libraries.
* Learn to preprocess the dataset, extract features, and apply feature selection techniques.

**Prerequisites:**

* Python installed (preferably using a virtual environment).
* Familiarity with libraries like pandas, numpy, scikit-learn, matplotlib, and seaborn.

**Step 1: Dataset Download and Setup**

1. **Dataset Selection**
   * Download a dataset - https://tinyurl.com/actwk6
   * For this exercise, you can use the **Iris dataset** (if you don’t have another dataset ready).

python

# Load the Iris dataset

from sklearn.datasets import load\_iris

import pandas as pd

# Load dataset into pandas DataFrame

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['target'] = iris.target

1. **Download a CSV Dataset** If you're using a CSV dataset (other than Iris), provide it to the students or have them download it, then load it using:

python

df = pd.read\_csv('path/to/your/dataset.csv')

**Step 2: Data Exploration**

1. **Inspect the Dataset** Encourage students to understand the dataset structure, check for missing values, and explore summary statistics.

python

# View the first few rows of the dataset

print(df.head())

# Check for missing values

print(df.isnull().sum())

# Summary statistics

print(df.describe())

1. **Data Visualization** Use visualizations to explore the relationships between features.

python

import seaborn as sns

import matplotlib.pyplot as plt

# Pairplot of features

sns.pairplot(df, hue='target')

plt.show()

**Step 3: Feature Extraction**

1. **Manual Feature Engineering**  
   Create new features based on the existing data. For example, creating polynomial features or combinations of columns.

python

# Create a new feature by combining existing features

df['sepal\_length\_to\_width'] = df['sepal length (cm)'] / df['sepal width (cm)']

1. **Extract Features Using Libraries** Extract numerical and categorical features. You can use libraries like sklearnfor automatic feature extraction.

python

Copy code

# Example using sklearn's PolynomialFeatures to generate polynomial features

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2, include\_bias=False)

features = df.drop(columns='target')

poly\_features = poly.fit\_transform(features)

print("Original features shape:", features.shape)

print("Polynomial features shape:", poly\_features.shape)

**Step 4: Feature Selection**

1. **Correlation Matrix** Use a correlation matrix to find features that are highly correlated and may need to be removed.

python

# Correlation matrix

corr\_matrix = df.corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.show()

1. **Using Variance Threshold** Remove features with low variance (they don’t provide much information).

python

from sklearn.feature\_selection import VarianceThreshold

# Select features based on variance

selector = VarianceThreshold(threshold=0.1)

selected\_features = selector.fit\_transform(features)

print("Selected features shape after variance threshold:", selected\_features.shape)

1. **Recursive Feature Elimination (RFE)** Use RFE to select the most important features based on a model.

python

from sklearn.feature\_selection import RFE

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

# Perform Recursive Feature Elimination

rfe = RFE(model, n\_features\_to\_select=3)

rfe = rfe.fit(features, df['target'])

print("Selected features (RFE):", rfe.support\_)

print("Feature ranking:", rfe.ranking\_)

1. **SelectKBest** Use SelectKBest to select the k best features using a scoring function.

python

from sklearn.feature\_selection import SelectKBest, f\_classif

# Select top 3 features based on ANOVA F-value between label and feature

selector = SelectKBest(f\_classif, k=3)

best\_features = selector.fit\_transform(features, df['target'])

print("Best features shape:", best\_features.shape)

**Step 5: Model Training with Selected Features**

1. **Train a Model** Use the selected features to train a model.

python

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(best\_features, df['target'], test\_size=0.3, random\_state=42)

# Train a simple Logistic Regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Make predictions and evaluate the model

predictions = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, predictions)

print(f"Model Accuracy with selected features: {accuracy:.4f}")

**Step 6: Conclusion**

* **Summarize Results:** After completing the steps, students should summarize their findings, discuss the importance of feature extraction and selection, and reflect on how selected features impacted model performance.