2. **Session 15: Feature Selection Part 2**
- Wrapper method
- Types of wrapper method
- Exhaustive Feature Selection/Best Subset Selection
- Sequential Backward Selection/Elimination
- Sequential Forward Selection
- Advantages and Disadvantages

3. **Session on Feature Selection part 3**
- Recursive Feature Elimination
- Advantages and Disadvantages

4. **PCA**
- Dimensionality reduction
- Scikit Learn's PCA class
- PCA variance Ratio
- Randomized PCA
- Incrimental PCA
- PCA on MNIST dataset
- Code with example

Wrapper Methods

Wrapper methods are a type of feature selection technique where a machine learning model is trained with different subsets of features, and the performance of the model determines which features contribute the most. They "wrap" around the machine learning model, iteratively testing feature subsets.

Types of Wrapper Methods:

1. Exhaustive Feature Selection (Best Subset Selection)

- **Concept:** Tests all possible combinations of features.
- **Mathematics:** If there are n features, the number of subsets is 2^n.
- **Code Example:**

python
Copy code

```
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
import pandas as pd

X, y = load_iris(return_X_y=True)
knn = KNeighborsClassifier(n_neighbors=3)
efs = EFS(knn, min_features=1, max_features=3, scoring='accuracy',
cv=5)
efs = efs.fit(X, y)

print('Best Features:', efs.best_feature_names_)
```

- **Advantages:**
  - Finds the optimal subset.
- **Disadvantages:**
  - Computationally expensive for large datasets.

2. Sequential Forward Selection (SFS)

- **Concept:** Starts with no features and adds one feature at a time based on performance improvement.
- **Mathematics:** Iteratively adds feature $XiXi$ if it maximizes a scoring function $J(X)J(X)$.
- **Code Example:**

python
Copy code

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
sfs = SFS(knn, k_features=3, forward=True, scoring='accuracy', cv=5)
sfs = sfs.fit(X, y)
print('Selected Features:', sfs.k_feature_names_)
```

- **Advantages:**
  - Less computational than exhaustive.
- **Disadvantages:**
  - Greedy approach; might miss optimal combinations.

3. Sequential Backward Selection (SBS)

- **Concept:** Starts with all features and removes the least contributing feature one by one.
- **Mathematics:** Iteratively removes feature $XiXi$ if it minimizes performance drop.
- **Code Example:**

python
Copy code

```
sbs = SFS(knn, k_features=3, forward=False, scoring='accuracy', cv=5)
sbs = sbs.fit(X, y)
print('Selected Features:', sbs.k_feature_names_)
```

Recursive Feature Elimination (RFE)

- **Concept:** Fits a model, ranks features based on importance, removes the least important, and repeats.
- **Mathematics:** Features are ranked based on coefficients or feature importance from models like SVM or Decision Trees.
- **Code Example:**

python
Copy code

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
rfe = RFE(model, n_features_to_select=3)
rfe.fit(X, y)
print('Selected Features:', rfe.support_)
```

- **Advantages:**
  - Good for feature ranking.
- **Disadvantages:**
  - Still computationally expensive.

Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that transforms the data into a new coordinate system where the features are linearly uncorrelated.

Mathematics of PCA:

1. **Standardize the Data:** $Xstandard=X-\mu\sigma X\text{standard}=\sigma X-\mu$
2. **Compute Covariance Matrix:** $C=1n-1XTXC=n-11\,XTX$
3. **Eigen Decomposition:** Solve $Cv=\lambda vCv=\lambda v$
4. **Project Data:** $Xprojected=XVkX\text{projected}=XVk$

Scikit-Learn PCA Example:

python
Copy code

```python
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.title('PCA Example on Iris')
plt.show()
```

Explained Variance Ratio:

python
Copy code

```python
print('Explained Variance Ratio:', pca.explained_variance_ratio_)
```

Randomized PCA (for large datasets):

python
Copy code

```python
pca_randomized = PCA(n_components=2, svd_solver='randomized')
X_pca_rand = pca_randomized.fit_transform(X)
```

Incremental PCA (for streaming data):

python

Copy code

```
from sklearn.decomposition import IncrementalPCA
ipca = IncrementalPCA(n_components=2, batch_size=10)
X_ipca = ipca.fit_transform(X)
```

PCA on MNIST Dataset:

python

Copy code

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784')
X_mnist, y_mnist = mnist.data, mnist.target

pca_mnist = PCA(n_components=100)
X_mnist_pca = pca_mnist.fit_transform(X_mnist)
print(f'Explained Variance by 100 components:
{sum(pca_mnist.explained_variance_ratio_)}')
```