ml_sem6_a1

March 30, 2024

[ ]:
```
## Aditya Agre
## 121B1B006
## ML assignment 1: Dimension Reduction
## 1. Select K Best : Select K best features out of given features.
```

[ ]:
```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2, r_regression
X, y = load_digits(return_X_y=True)
X.shape
```

[ ]: (1797, 64)

[ ]:
```
print(X[0:2])
```

```
[[ 0.  0.  5. 13.  9.  1.  0.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.
  15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.
   0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.
   0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]
 [ 0.  0.  0. 12. 13.  5.  0.  0.  0.  0.  0. 11. 16.  9.  0.  0.  0.  0.
   3. 15. 16.  6.  0.  0.  0.  7. 15. 16. 16.  2.  0.  0.  0.  0.  1. 16.
  16.  3.  0.  0.  0.  0.  1. 16. 16.  6.  0.  0.  0.  0.  1. 16. 16.  6.
   0.  0.  0.  0.  0. 11. 16. 10.  0.  0.]]
```

[ ]:
```
print(y[0:2])
```

```
[0 1]
```

[ ]:
```
## Using Karl Pearson coefficient as score function.
## We are using Karl Pearson coeff because, both X features and target y are␣
 ↪continnuous.
## selecting top 20 features (k = 20)
X_new = SelectKBest(r_regression, k=20).fit_transform(X, y)
```

[ ]:
```
X_new.shape
```

[ ]: (1797, 20)

[ ]:
```
## Therefore, we have selected 20 of the best features out of the given 64␣
 ↪features.
```

```
## 2. Select Percentile Method : Select features according to a percentile of the highest scores.
## this time, let's use chi as the score function.
## Compute Pearson's r for each features and the target.
## Pearson's r is also known as the Pearson correlation coefficient.
```

[ ]:
```
from sklearn.feature_selection import SelectPercentile, chi2
```

[ ]:
```
## selecting into X_new only the top 10 percentile of features with the best scores acc to chi score funtion.
X_new = SelectPercentile(chi2, percentile=10).fit_transform(X, y)
```

[ ]:
```
## !0 percentile of 64 equal tpo 64/10. Therefore top 10 features taken.
X_new.shape
```

[ ]:
```
(1797, 7)
```

[ ]:
```
## 3. PCA: pRINCIPAL cOMPONENT aNALYSIS
```

[ ]:
```
from sklearn.decomposition import PCA
import numpy as np
```

[ ]:
```
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
pca = PCA(n_components=2)
pca.fit(X)
```

[ ]:
```
PCA(n_components=2)
```

[ ]:
```
print(pca.explained_variance_ratio_)
```

```
[0.99244289 0.00755711]
```

[ ]:
```
print(pca.singular_values_)
```

```
[6.30061232 0.54980396]
```

[ ]:
```

```