ML - EXP - 8

Atharva Prashant Pawar (9427) - [Batch - D]

PCA : Principal Component Analysis

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

np.random.seed(23)

mu_vec1 = np.array([0,0,0])
cov_mat1 = np.array([[1,0,0],[0,1,0],[0,0,1]])
class1_sample = np.random.multivariate_normal(mu_vec1, cov_mat1, 20)

df = pd.DataFrame(class1_sample,columns=['feature1','feature2','feature3'])
df['target'] = 1

mu_vec2 = np.array([1,1,1])
cov_mat2 = np.array([[1,0,0],[0,1,0],[0,0,1]])
class2_sample = np.random.multivariate_normal(mu_vec2, cov_mat2, 20)

df1 = pd.DataFrame(class2_sample,columns=['feature1','feature2','feature3'])

df1['target'] = 0

df = df.append(df1,ignore_index=True)

df = df.sample(40)
```

```
<ipython-input-24-b42d3f200777>:23: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
```

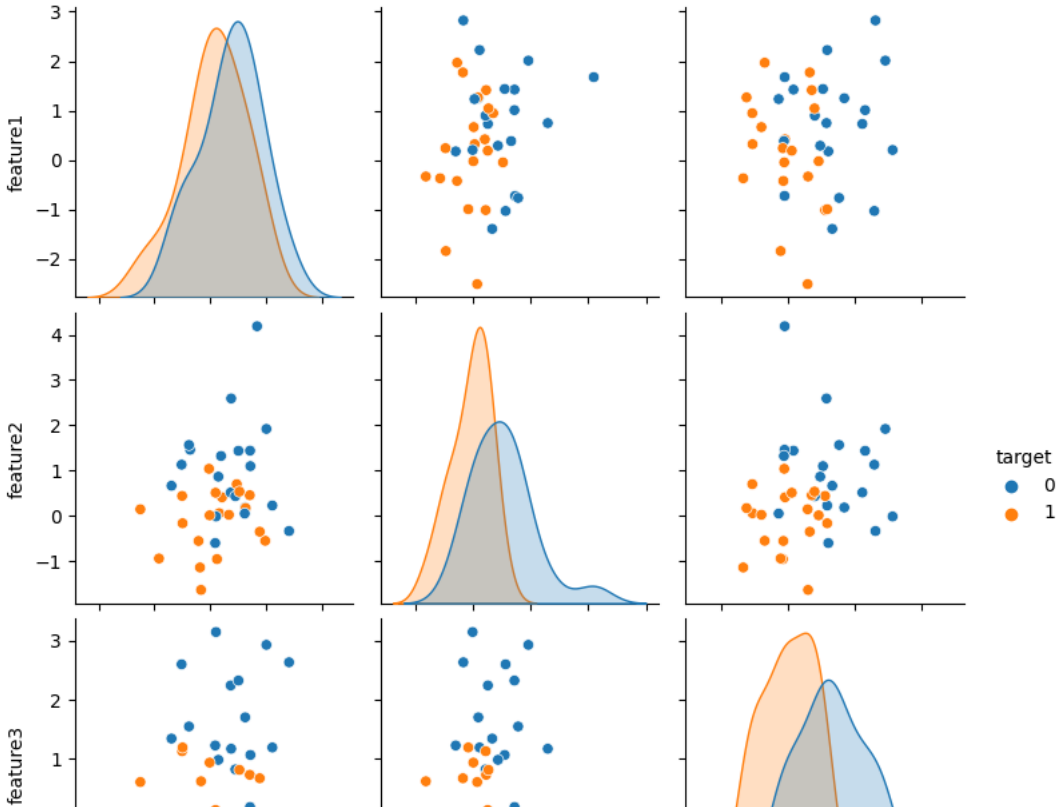
```
df.head()
```

	feature1	feature2	feature3	target
2	-0.367548	-1.137460	-1.322148	1
34	0.177061	-0.598109	1.226512	0
14	0.420623	0.411620	-0.071324	1
11	1.968435	-0.547788	-0.679418	1
12	-2.506230	0.146960	0.606195	1

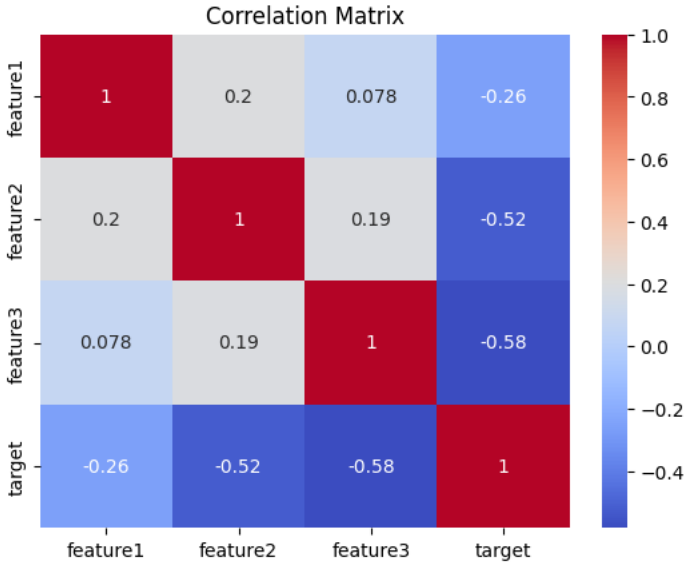
```
# Summary statistics
print("Summary Statistics:")
print(df.describe())
```

```
Summary Statistics:
        feature1   feature2   feature3    target
count  40.000000  40.000000  40.000000  40.00000
mean    0.433721   0.460790   0.667670   0.50000
std     1.157915   1.060976   1.152079   0.50637
min    -2.506230  -1.632386  -1.322148   0.00000
25%    -0.340600  -0.048988  -0.107669   0.00000
50%     0.402744   0.423790   0.699508   0.50000
75%     1.254864   1.055595   1.200931   1.00000
max     2.823378   4.187503   3.150780   1.00000
```

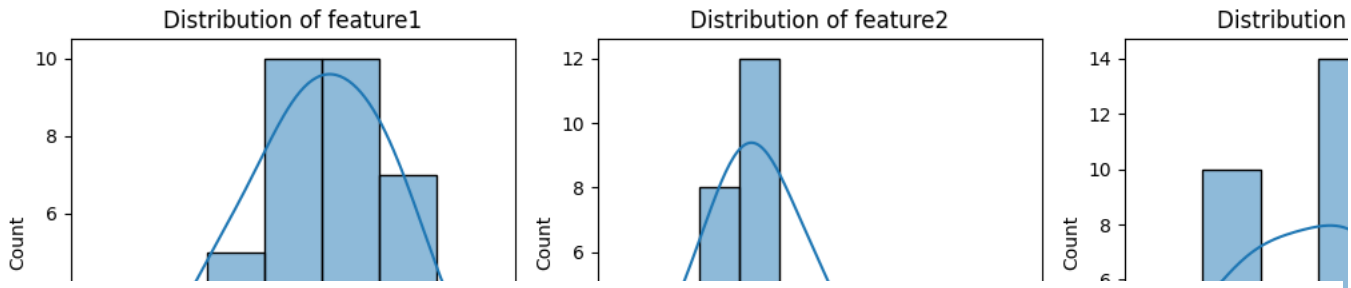
```
# Pairplot
sns.pairplot(df, hue='target', diag_kind='kde')
plt.show()
```




```
# Correlation matrix
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```




```
# Distribution of each feature
plt.figure(figsize=(12, 4))
for i, feature in enumerate(['feature1', 'feature2', 'feature3']):
    plt.subplot(1, 3, i + 1)
    sns.histplot(df[feature], kde=True)
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
```




```
import plotly.express as px
#y_train_trf = y_train.astype(str)
fig = px.scatter_3d(df, x=df['feature1'], y=df['feature2'], z=df['feature3'],
              color=df['target'].astype('str'))
fig.update_traces(marker=dict(size=12,
                              line=dict(width=2,
                                        color='DarkSlateGrey')),
                  selector=dict(mode='markers'))

fig.show()
```


```
# Step 1 - Apply standard scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

df.iloc[:,0:3] = scaler.fit_transform(df.iloc[:,0:3])


# Step 2 - Find Covariance Matrix
covariance_matrix = np.cov([df.iloc[:,0],df.iloc[:,1],df.iloc[:,2]])
print('Covariance Matrix:\n', covariance_matrix)
```

```
Covariance Matrix:
 [[1.02564103 0.20478114 0.080118  ]
 [0.20478114 1.02564103 0.19838882]
 [0.080118   0.19838882 1.02564103]]
```

```
# Step 3 - Finding EV and EVs
eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)
```



```
print("eigen_values : ", eigen_values , "\n", "eigen_vectors : ", eigen_vectors , "\n")
```

```
eigen_values :  [1.3536065  0.94557084 0.77774573]
 eigen_vectors :  [[-0.53875915 -0.69363291  0.47813384]
 [-0.65608325 -0.01057596 -0.75461442]
 [-0.52848211  0.72025103  0.44938304]]
```

```
pc = eigen_vectors[0:2]
pc
```

```
array([[-0.53875915, -0.69363291,  0.47813384],
       [-0.65608325, -0.01057596, -0.75461442]])
```

```
transformed_df = np.dot(df.iloc[:,0:3],pc.T)
# 40,3 - 3,2
new_df = pd.DataFrame(transformed_df,columns=['PC1','PC2'])
new_df['target'] = df['target'].values
new_df.head()
```

	PC1	PC2	target
0	0.354836	1.250883	1
1	0.905912	-1.035385	0
2	-0.546230	-0.226495	1
3	-1.005401	-0.772965	1
4	1.538160	1.185298	1

```
new_df['target'] = new_df['target'].astype('str')
fig = px.scatter(x=new_df['PC1'],
                 y=new_df['PC2'],
                 color=new_df['target'],
                 color_discrete_sequence=px.colors.qualitative.G10
                )

fig.update_traces(marker=dict(size=12,
                              line=dict(width=2,
                                        color='DarkSlateGrey')),
                  selector=dict(mode='markers'))
fig.show()
```