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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
df = pd.read_csv("/content/drive/MyDrive/Intro ML 2022 Summer/dataset/Social_Network_
```

```
df.head()
```

```
df.info()
```

```
df.describe()
```

```
X=df[['Age','EstimatedSalary']]
y = df['Purchased']
```

```
Xarr=X.to_numpy()
```

```
yarr=y.to_numpy()
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(Xarr)
```

```
plt.boxplot(X)
plt.show()
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train,y_test = train_test_split(X,yarr,test_size = 0.2)
X.shape, X_train.shape, X_test.shape
```

```
from sklearn.svm import SVC

classifier = SVC(kernel='linear', C=1)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_pred
```

```
from sklearn.metrics import confusion_matrix
compare = confusion_matrix(y_test, y_pred)
compare
```

```
accuracy = (compare[0][0]+compare[1][1])/compare.sum()
accuracy
```

```
from sklearn.metrics import accuracy_score

test_acc = accuracy_score(y_test, y_pred)
test_acc
```

```
print(classifier.predict([[1, -1.5]]))
```

```
print(classifier.predict([[1, 0]]))
```

## Next: Plot a scatter plot with the decision boundary and the parallels

```
def plot_support_vector_machine(svm):

    plt.scatter(X[:, 0], X[:, 1], c=yarr, zorder=10, cmap=plt.cm.Paired,
                edgecolor='k', s=20)

    plt.scatter(X_test[:, 0], X_test[:, 1], s=100, facecolors='none',
                zorder=10, edgecolor='k')

    plt.axis('tight')
    x_min = X[:, 0].min()
    x_max = X[:, 0].max()
    y_min = X[:, 1].min()
    y_max = X[:, 1].max()

    XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
    Z = svm.decision_function(np.c_[XX.ravel(), YY.ravel()])

    Z = Z.reshape(XX.shape)
    plt.pcolormesh(XX, YY, Z > 0, cmap=plt.cm.Paired, shading="auto")
    plt.contour(XX, YY, Z, colors=['k', 'k', 'k'],
                linestyles=['--', '-', '--'], levels=[-.5, 0, .5])
```

```
plt.figure(figsize=(7,7))
plot_support_vector_machine(classifier)
```

## Poly SVM and RBF

```
poly_svc = SVC(kernel="poly")
poly_svc.fit(X_train, y_train)

y_pred = poly_svc.predict(X_test)
y_pred
```

```
plt.figure(figsize=(7,7))
plot_support_vector_machine(poly_svc)
```

```
from sklearn.metrics import accuracy_score

test_acc = accuracy_score(y_test, y_pred)
test_acc
```

```
rbf_svc = SVC(kernel="rbf", gamma=100, C=1)
rbf_svc.fit(X_train, y_train)

y_pred = rbf_svc.predict(X_test)
y_pred
```

```
plt.figure(figsize=(7, 7))
plot_support_vector_machine(rbf_svc)
```

```
from sklearn.metrics import accuracy_score

test_acc = accuracy_score(y_test, y_pred)
test_acc
```

## Parameter tuning

```
from sklearn.model_selection import GridSearchCV
```

```
params = [
    {"kernel": ["linear"], "C": [0.1, 0.5, 1, 5, 10, 30]},
    {"kernel": ["poly"],"C": [0.1, 0.5, 1, 5, 10, 30] },
    {"kernel": ["rbf"], "C": [0.1, 0.5, 1, 5, 10, 30],
                      "gamma": [0.01, 0.03, 0.1, 0.3, 1.0, 3.0,10.0],},]
```

```
grid_cv = GridSearchCV(classifier, params, cv=5,n_jobs=-1)
grid_cv.fit(X_train, y_train)
```

```
print(f"Highest score of paramter search is: {grid_cv.best_score_:.4f}")
```

```
print("The parameter of the highest score is as follows")
for key, value in grid_cv.best_params_.items():
    print(f"{key}: {value}")
```

```
best_test_acc = accuracy_score(y_test, y_pred)
```

```
print(f"Highest parameter test accuracy is {best_test_acc:.3f}")
```

✓ 1s completed at 11:31 AM ● ✕