In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

In [ ]:
```
dataset = pd.read_csv('../Social_Network_Ads.csv')
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['Gender']=le.fit_transform(dataset['Gender'])
```

In [ ]:
```
print(dataset.head())
```

```
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510       1   19            19000          0
1  15810944       1   35            20000          0
2  15668575       0   26            43000          0
3  15603246       0   27            57000          0
4  15804002       1   19            76000          0
```

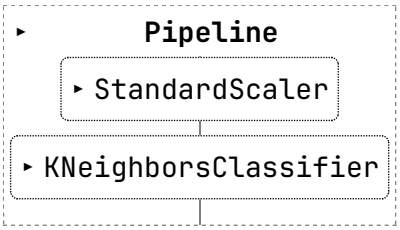
In [ ]:
```
X = dataset.iloc[:, 2:4].values
y = dataset.iloc[:, -1].values
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size = 0.25, random_state = 0)
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
```

In [ ]:
```
knn = Pipeline([
    ('scaler', StandardScaler()),
    ('knn', KNeighborsClassifier(n_neighbors=11))
])
knn.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```
y_pred = knn.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

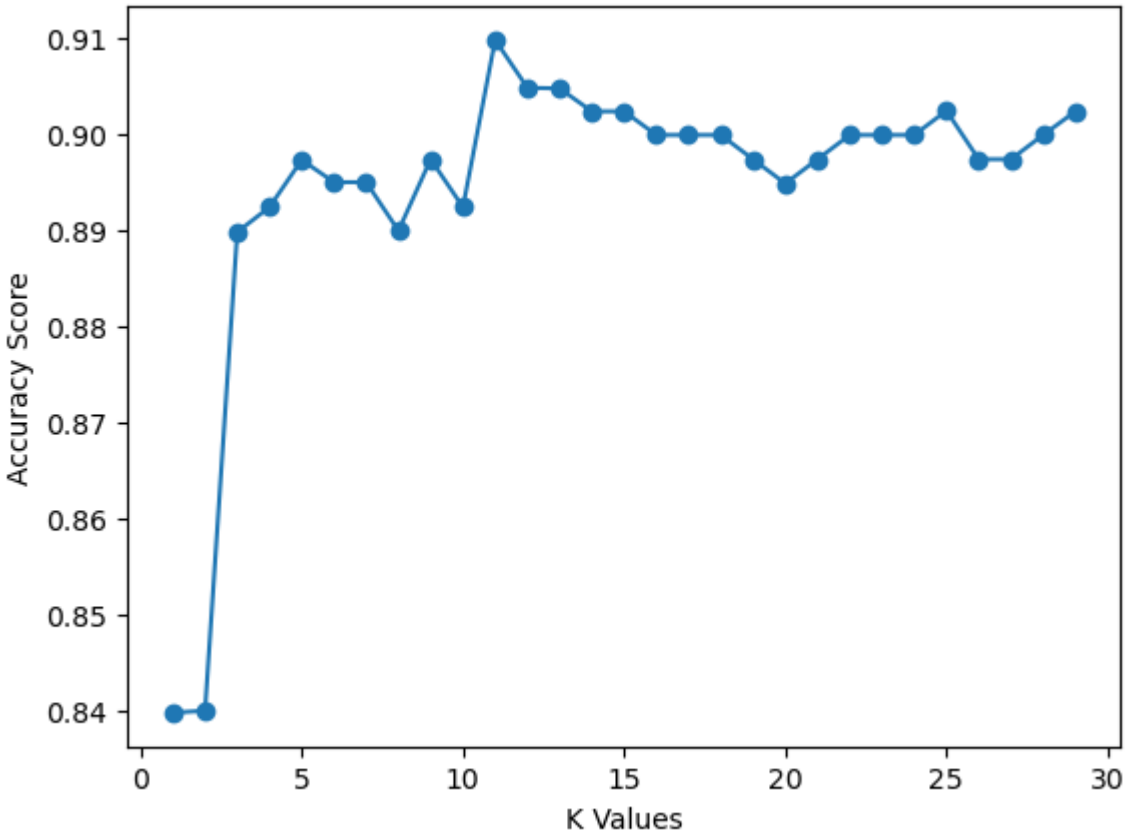
```
Accuracy: 0.93
```

In [ ]:
```
from sklearn.model_selection import  cross_val_score
k_values = [i for i in range (1,30)]
scores = []

scaler = StandardScaler()
X_ = scaler.fit_transform(X)
```

```
for k_ in k_values:
    knn_ = KNeighborsClassifier(n_neighbors=k_)
    score = cross_val_score(knn_, X_, y, cv=11)
    scores.append(np.mean(score))
```

In [ ]:
```
plt.plot(k_values, scores, marker='o')
plt.xlabel("K Values")
plt.ylabel("Accuracy Score")
plt.show()
```

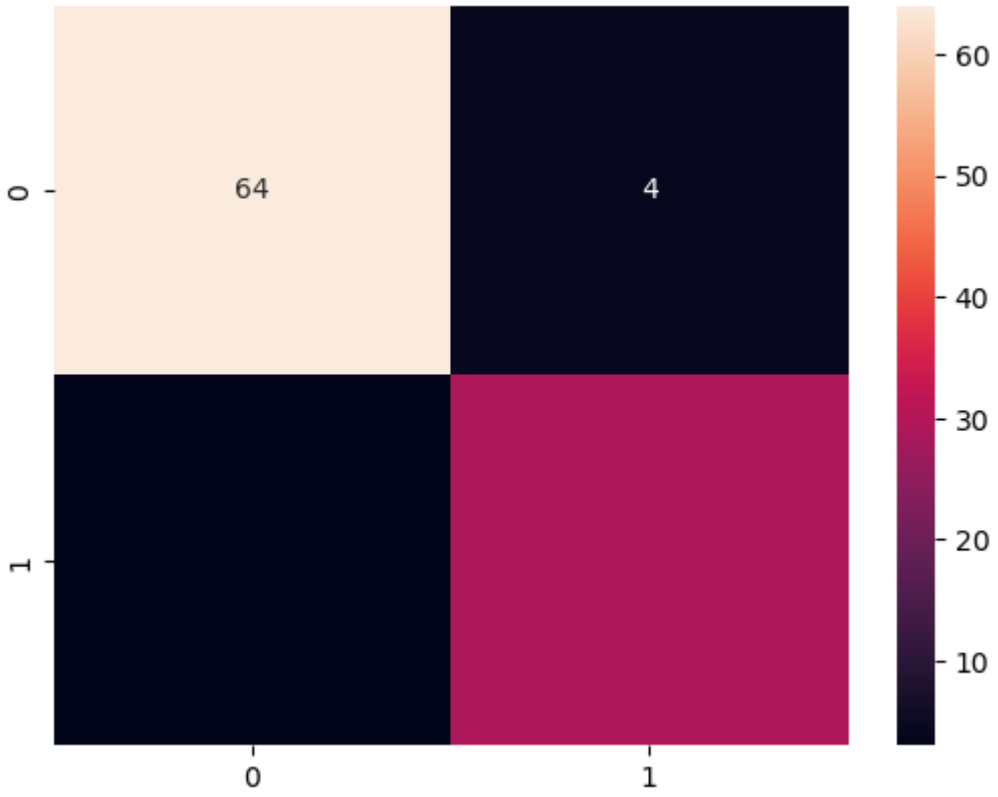


In [ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[64  4]
 [ 3 29]]
```

Out[ ]: 0.93

In [ ]:
```
import seaborn as sns
sns.heatmap(cm, annot=True, fmt="d")
plt.show()
```



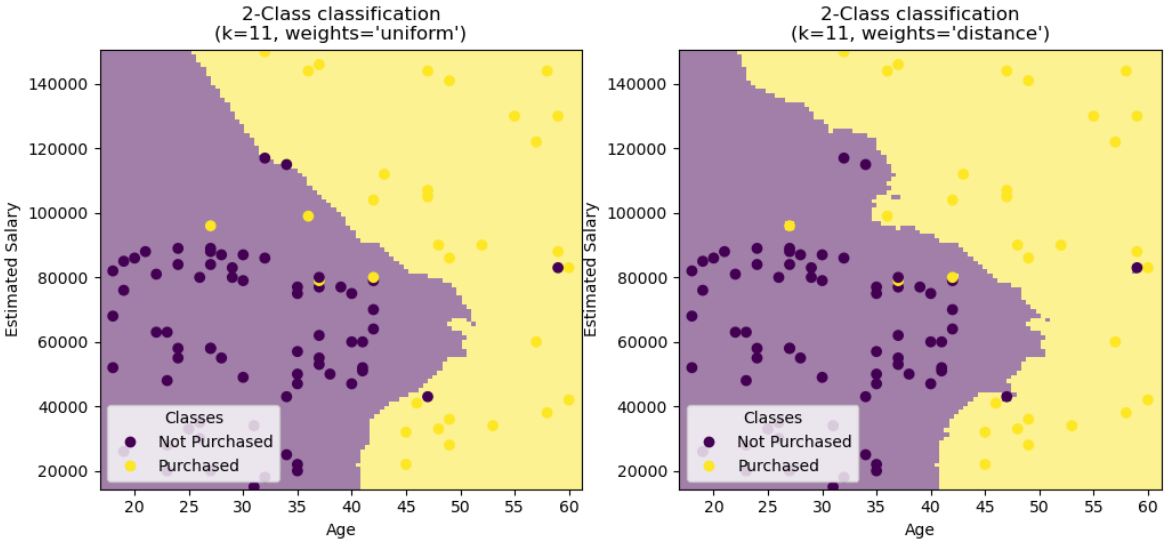
In [ ]:

```
from sklearn.inspection import DecisionBoundaryDisplay

_, axs = plt.subplots(ncols=2, figsize=(12, 5))
label_mapping = {0: "Not Purchased", 1: "Purchased"}

for ax, weights in zip(axs, ("uniform", "distance")):
    knn.set_params(knn__weights=weights).fit(X_test[:, 0:2], y_test)
    disp = DecisionBoundaryDisplay.from_estimator(
        knn,
        X_test[:, 0:2],
        response_method="predict",
        plot_method="pcolormesh",
        ylabel="Estimated Salary",
        xlabel="Age",
        shading="auto",
        alpha=0.5,
        ax=ax,
    )
    scatter = disp.ax_.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
    disp.ax_.legend(
        scatter.legend_elements()[0],
        [label_mapping[int(label)] for label in dataset['Purchased'].uniqu
        loc="lower left",
        title="Classes",
    )
    _ = disp.ax_.set_title(
        f"2-Class classification\n(k=11, weights={weights!r})"
    )

plt.show()
```


2-Class classification
(k=11, weights='uniform')
2-Class classification
(k=11, weights='distance')
Estimated Salary
Age
Classes
Not Purchased
Purchased
140000
120000
100000
80000
60000
40000
20000
20
25
30
35
40
45
50
55
60