Youssef Mahmoud 905854027.

In [1]:

```
import numpy as np
import pandas as pd
import patsy as pt
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, roc_auc_score

sns.set()
```

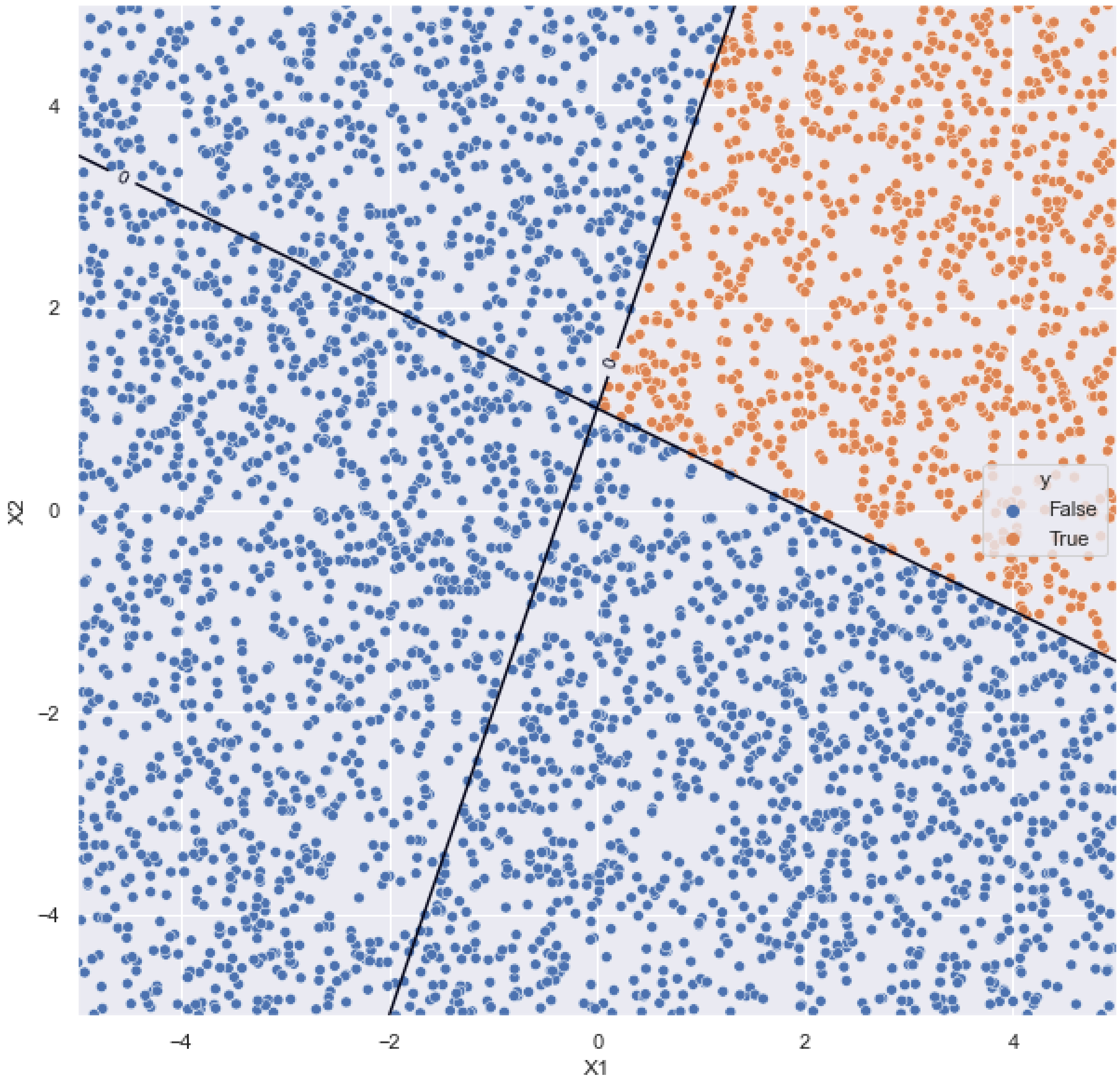
9.7.1 A & B

In [2]:

```
x1 = np.linspace(-5.0, 5.0, 100)
x2 = np.linspace(-5.0, 5.0, 100)
X1, X2 = np.meshgrid(x1,x2)
Y1 = 1 + 3*X1 -X2
Y2 = -2 + X1 + 2*X2


x1 = np.random.uniform(-1, 1, 4000) * 5
x2 = np.random.uniform(-1, 1, 4000) * 5
y1 = ((1 + 3*x1 -x2) > 0)*1
y2 = ((-2 + x1 + 2*x2) > 0)*1
y = np.all([y1, y2], axis=0)

df = pd.DataFrame({'x1':x1, 'x2':x2, 'y':y})


fig, ax = plt.subplots(figsize=(10,10))
CS1 = ax.contour(X1,X2,Y1, [0])
ax.clabel(CS1, inline=1, fontsize=10)
CS2 = ax.contour(X1,X2,Y2, [0])
ax.clabel(CS2, inline=1, fontsize=10)
sns.scatterplot(x='x1', y='x2', hue='y', data=df)
plt.xlabel('X1')
plt.ylabel('X2')

```

Out[2]:

```
Text(0, 0.5, 'X2')
```

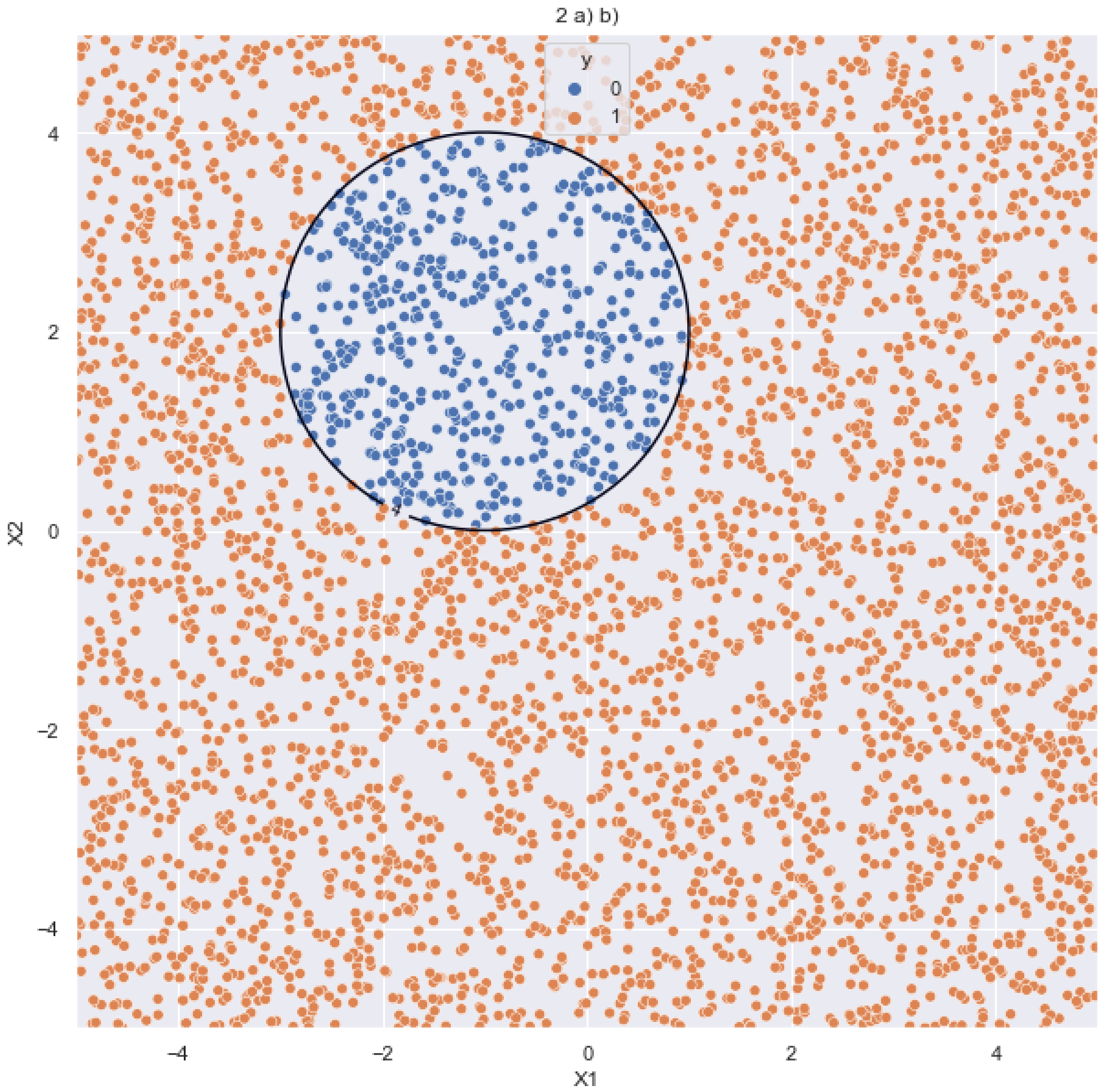

0
0
y
● False
● True
4
2
0
−2
−4
X2
−4
−2
0
2
4
X1

9.7.2

A & B

In [3]:

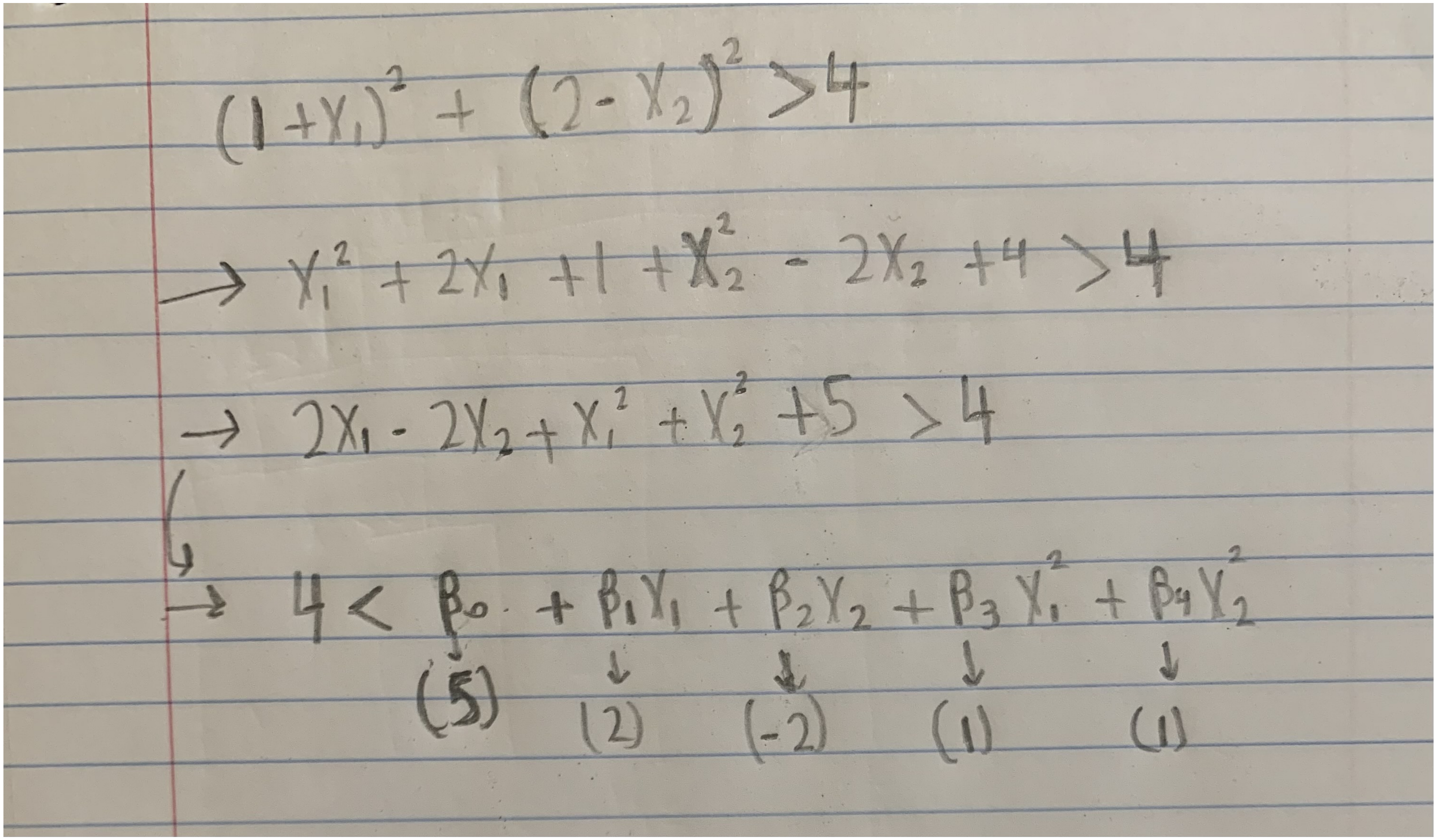
```
np.random.seed(0)

# Gen samples
x1 = np.random.uniform(-1, 1, 4000) * 5
x2 = np.random.uniform(-1, 1, 4000) * 5
y = 1*(((1+x1)**2 + (2-x2)**2) > 4)
df = pd.DataFrame({'x1':x1, 'x2':x2, 'y':y})

# Draw decision line
x1 = np.linspace(-5.0, 5.0, 100)
x2 = np.linspace(-5.0, 5.0, 100)
X1, X2 = np.meshgrid(x1,x2)
Y = (1+X1)**2 + (2-X2)**2

# Plot data
fig, ax = plt.subplots(figsize=(10,10))
CS = ax.contour(X1,X2,Y, [4])
sns.scatterplot(x='x1', y='x2', hue='y', data=df)
ax.clabel(CS, inline=1, fontsize=10)
plt.xlabel('X1')
plt.ylabel('X2')
ax.set_title('2 a) b)');
```



C

(0, 0) is orange, (−1, 1) is blue, (2, 2) is orange, (3, 8) is orange

D



$$(1+X_1)^2 + (2-X_2)^2 > 4$$

$$\rightarrow X_1^2 + 2X_1 + 1 + X_2^2 - 2X_2 + 4 > 4$$

$$\rightarrow 2X_1 - 2X_2 + X_1^2 + X_2^2 + 5 > 4$$

$$\rightarrow 4 < \underset{\substack{\downarrow \\ (5)}}{\beta_0} + \underset{\substack{\downarrow \\ (2)}}{\beta_1} X_1 + \underset{\substack{\downarrow \\ (-2)}}{\beta_2} X_2 + \underset{\substack{\downarrow \\ (1)}}{\beta_3} X_1^2 + \underset{\substack{\downarrow \\ (1)}}{\beta_4} X_2^2$$

SVMs

In [4]:

```
df = pd.read_csv("Desktop/Cars.csv")
```

In [5]:

```
df
```

Out[5]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No
...	...	...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	Good	33	14	Yes	Yes
396	6.14	139	23	3	37	120	Medium	55	11	No	Yes
397	7.41	162	26	12	368	159	Medium	40	18	Yes	Yes
398	5.94	100	79	7	284	95	Bad	50	12	Yes	Yes
399	9.71	134	37	0	27	120	Good	49	16	Yes	Yes

400 rows × 11 columns

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Sales        400 non-null    float64
 1   CompPrice    400 non-null    int64  
 2   Income       400 non-null    int64  
 3   Advertising  400 non-null    int64  
 4   Population   400 non-null    int64  
 5   Price        400 non-null    int64  
 6   ShelveLoc    400 non-null    object 
 7   Age          400 non-null    int64  
 8   Education    400 non-null    int64  
 9   Urban        400 non-null    object 
 10  US           400 non-null    object 
dtypes: float64(1), int64(7), object(3)
memory usage: 34.5+ KB
```

In [7]:

```
df['Sales1'] = df.Sales.map(lambda x: 1 if x>8 else 0)
```

In [8]:

```
df
```

Out[8]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US	Sales1
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes	1
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes	1
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes	1
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes	0
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No	0
...	...	...	...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	Good	33	14	Yes	Yes	1
396	6.14	139	23	3	37	120	Medium	55	11	No	Yes	0
397	7.41	162	26	12	368	159	Medium	40	18	Yes	Yes	0
398	5.94	100	79	7	284	95	Bad	50	12	Yes	Yes	0
399	9.71	134	37	0	27	120	Good	49	16	Yes	Yes	1

400 rows × 12 columns

In [9]:

```
X1 = df.drop(['Sales', 'Sales1'], axis=1)
Y1 = df['Sales1']
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size=0.25, random_state=425
```

In [10]:

```
num_features = ['CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age', 'Education']
cat_features = ['Urban', 'US','ShelveLoc']
features = np.concatenate([num_features, cat_features])
```

In [11]:

```
categorical_tf = Pipeline(steps = [
  ("cat_impute", SimpleImputer(strategy = 'most_frequent')),
  ("encoder", OneHotEncoder())
])

# Transformer for continuous variables
numeric_tf = Pipeline(steps = [
  ("num_impute", SimpleImputer(strategy = 'mean')),
])

# Column transformer
col_tf = ColumnTransformer(transformers = [
  ('num', numeric_tf, num_features),
  ('cat', categorical_tf, cat_features)
])
```

Linear Kernel

In [12]:

```
svm_mod = SVC(
  C = 1.0,
  kernel = 'linear',
  gamma = 'scale',
  probability = True,
  random_state = 425
  )
```

In [13]:

```
pipe = Pipeline(steps = [
  ("col_tf", col_tf),
  ("model", svm_mod)
  ])
pipe
```

Out[13]:

```
Pipeline(steps=[('col_tf',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_impute',
                                                                   SimpleImputer())]),
                                                  ['CompPrice', 'Income',
                                                   'Advertising', 'Population',
                                                   'Price', 'Age',
                                                   'Education']),
                                                 ('cat',
                                                  Pipeline(steps=[('cat_impute',
                                                                   SimpleImputer(strategy='most_fr
equent')),
                                                                  ('encoder',
                                                                   OneHotEncoder())]),
                                                  ['Urban', 'US',
                                                   'ShelveLoc'])])),
                ('model',
                 SVC(kernel='linear', probability=True, random_state=425))])
```

In [14]:

```
C_grid = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]
gamma_grid = np.logspace(start = -3, stop = 0, base = 10, num = 10)
tuned_parameters = {
  "model__C": C_grid,
  "model__gamma": gamma_grid
  }
tuned_parameters
```

Out[14]:

```
{'model__C': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0],
 'model__gamma': array([0.001     , 0.00215443, 0.00464159, 0.01      , 0.02154435,
        0.04641589, 0.1       , 0.21544347, 0.46415888, 1.        ])}
```

In [15]:

```
n_folds = 5
search = GridSearchCV(
  pipe,
  tuned_parameters,
  cv = n_folds,
  scoring = "roc_auc",
  refit = True
  )
```

In [16]:

```
search.fit(X1_train, Y1_train)

```

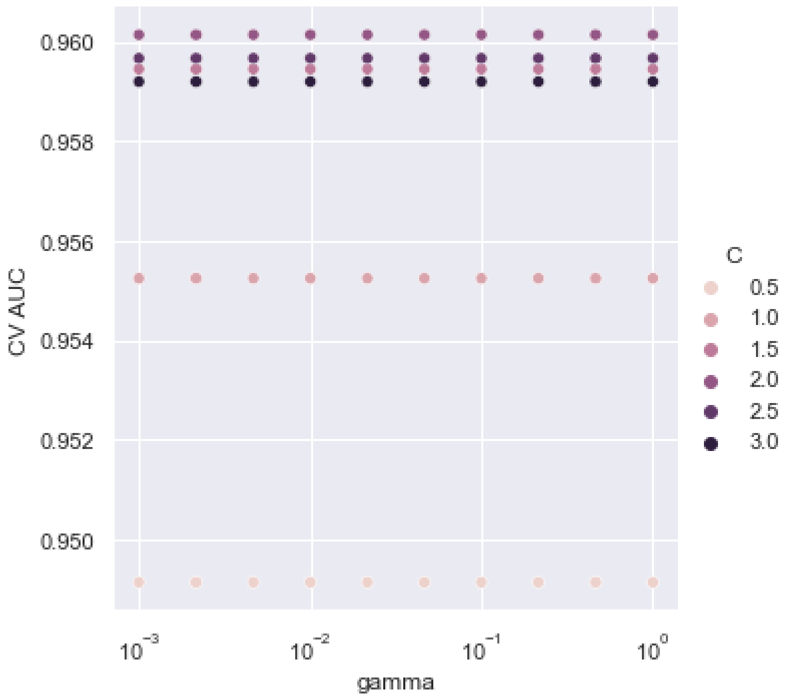
Out[16]:

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('col_tf',
                                        ColumnTransformer(transformers=[('num',
                                                                         Pipeline(steps=[('num_imp
ute',
                                                                                          SimpleIm
puter())]),
                                                                         ['CompPrice',
                                                                          'Income',
                                                                          'Advertising',
                                                                          'Population',
                                                                          'Price',
                                                                          'Age',
                                                                          'Education']),
                                                                        ('cat',
                                                                         Pipeline(steps=[('cat_imp
ute',
                                                                                          SimpleIm
puter(strategy='most_frequent')),
                                                                                         ('encode
r',
                                                                                          OneHotEn
coder())]),
                                                                         ['Urban',
                                                                          'US',
                                                                          'ShelveLoc'])])),
                                       ('model',
                                        SVC(kernel='linear', probability=True,
                                            random_state=425))]),
             param_grid={'model__C': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0],
                         'model__gamma': array([0.001     , 0.00215443, 0.00464159, 0.01      , 0.
02154435,
       0.04641589, 0.1       , 0.21544347, 0.46415888, 1.        ])},
             scoring='roc_auc')
```

In [23]:

```
cv_res = pd.DataFrame({
  "C": np.array(search.cv_results_["param_model__C"]),
  "auc": search.cv_results_["mean_test_score"],
  "gamma": search.cv_results_["param_model__gamma"]
  })

plt.figure()
sns.relplot(
  # kind = "line",
  data = cv_res,
  x = "gamma",
  y = "auc",
  hue = "C"
  ).set(
    xscale = "log",
    xlabel = "gamma",
    ylabel = "CV AUC"
    );
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [24]:

```
search.best_score_

```

Out[24]:

```
0.9601389432911173
```

In [25]:

```
accuracy_score(
  Y1_train,
  search.best_estimator_.predict(X1_train)
  )
```

Out[25]:

```
0.9333333333333333
```

In [26]:

```
search.best_estimator_

```

Out[26]:

```
Pipeline(steps=[('col_tf',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_impute',
                                                                   SimpleImputer())]),
                                                  ['CompPrice', 'Income',
                                                   'Advertising', 'Population',
                                                   'Price', 'Age',
                                                   'Education']),
                                                 ('cat',
                                                  Pipeline(steps=[('cat_impute',
                                                                   SimpleImputer(strategy='most_fr
equent')),
                                                                  ('encoder',
                                                                   OneHotEncoder())]),
                                                  ['Urban', 'US',
                                                   'ShelveLoc'])])),
                ('model',
                 SVC(C=2.0, gamma=0.001, kernel='linear', probability=True,
                     random_state=425))])
```

In [27]:

```
roc_auc_score(
  Y1_test,
  search.best_estimator_.predict_proba(X1_test)[:, 1]
  )
```

Out[27]:

```
0.9281413087113609
```

In [28]:

```
accuracy_score(
  Y1_test,
  search.best_estimator_.predict(X1_test)
  )
```

Out[28]:

```
0.84
```

radial kernel

In [29]:

```python
svm_mod = SVC(
  C = 1.0,
  kernel = 'rbf',
  gamma = 'scale', # 1 / (n_features * X.var())
  probability = True,
  random_state = 425
  )
```

In [30]:

```python
pipe = Pipeline(steps = [
  ("col_tf", col_tf),
  ("model", svm_mod)
  ])
pipe
```

Out[30]:

```
Pipeline(steps=[('col_tf',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_impute',
                                                                   SimpleImputer())]),
                                                  ['CompPrice', 'Income',
                                                   'Advertising', 'Population',
                                                   'Price', 'Age',
                                                   'Education']),
                                                 ('cat',
                                                  Pipeline(steps=[('cat_impute',
                                                                   SimpleImputer(strategy='most_fr
equent')),
                                                                  ('encoder',
                                                                   OneHotEncoder())]),
                                                  ['Urban', 'US',
                                                   'ShelveLoc'])])),
                ('model', SVC(probability=True, random_state=425))])
```

In [31]:

```
C_grid = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]
gamma_grid = np.logspace(start = -3, stop = 0, base = 10, num = 10)
tuned_parameters = {
  "model__C": C_grid,
  "model__gamma": gamma_grid
  }
tuned_parameters
```

Out[31]:

```
{'model__C': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0],
 'model__gamma': array([0.001     , 0.00215443, 0.00464159, 0.01      , 0.02154435,
       0.04641589, 0.1       , 0.21544347, 0.46415888, 1.        ])}
```

In [32]:

```
n_folds = 5
search = GridSearchCV(
  pipe,
  tuned_parameters,
  cv = n_folds,
  scoring = "roc_auc",
  # Refit the best model on the whole data set
  refit = True
  )
```

In [33]:

```
search.fit(X1_train, Y1_train)

```

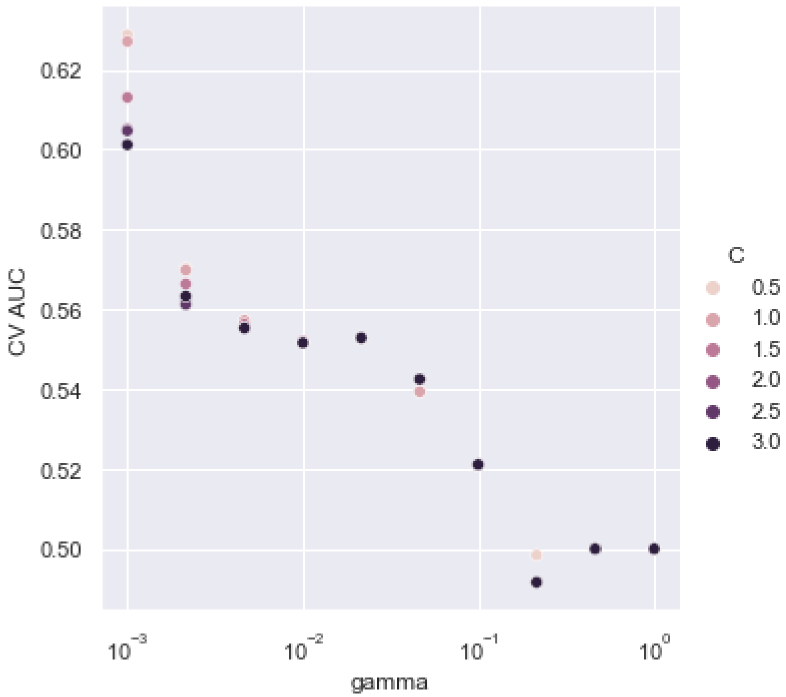
Out[33]:

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('col_tf',
                                        ColumnTransformer(transformers=[('num',
                                                                         Pipeline(steps=[('num_imp
ute',
                                                                                          SimpleIm
puter())]),
                                                                         ['CompPrice',
                                                                          'Income',
                                                                          'Advertising',
                                                                          'Population',
                                                                          'Price',
                                                                          'Age',
                                                                          'Education']),
                                                                        ('cat',
                                                                         Pipeline(steps=[('cat_imp
ute',
                                                                                          SimpleIm
puter(strategy='most_frequent')),
                                                                                         ('encode
r',
                                                                                          OneHotEn
coder())]),
                                                                         ['Urban',
                                                                          'US',
                                                                          'ShelveLoc'])])),
                                       ('model',
                                        SVC(probability=True,
                                            random_state=425))]),
             param_grid={'model__C': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0],
                         'model__gamma': array([0.001     , 0.00215443, 0.00464159, 0.01      , 0.
02154435,
       0.04641589, 0.1       , 0.21544347, 0.46415888, 1.        ])},
             scoring='roc_auc')
```

In [34]:

```
cv_res = pd.DataFrame({
  "C": np.array(search.cv_results_["param_model__C"]),
  "auc": search.cv_results_["mean_test_score"],
  "gamma": search.cv_results_["param_model__gamma"]
  })

plt.figure()
sns.relplot(
  # kind = "line",
  data = cv_res,
  x = "gamma",
  y = "auc",
  hue = "C"
  ).set(
    xscale = "log",
    xlabel = "gamma",
    ylabel = "CV AUC"
    );
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [35]:

```
search.best_score_

```

Out[35]:

```
0.6288059798929363
```

In [36]:

```
accuracy_score(
  Y1_train,
  search.best_estimator_.predict(X1_train)
  )
```

Out[36]:

```
0.7533333333333333
```

In [37]:

```
search.best_estimator_

```

Out[37]:

```
Pipeline(steps=[('col_tf',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_impute',
                                                                   SimpleImputer())]),
                                                  ['CompPrice', 'Income',
                                                   'Advertising', 'Population',
                                                   'Price', 'Age',
                                                   'Education']),
                                                 ('cat',
                                                  Pipeline(steps=[('cat_impute',
                                                                   SimpleImputer(strategy='most_fr
equent')),
                                                                  ('encoder',
                                                                   OneHotEncoder())]),
                                                  ['Urban', 'US',
                                                   'ShelveLoc'])])),
                ('model',
                 SVC(C=0.5, gamma=0.001, probability=True, random_state=425))])
```

In [39]:

```
roc_auc_score(
  Y1_test,
  search.best_estimator_.predict_proba(X1_test)[:, 1]
  )
```

Out[39]:

```
0.6724207145724608
```

In [40]:

```
accuracy_score(
  Y1_test,
  search.best_estimator_.predict(X1_test)
  )
```

Out[40]:

```
0.55
```

Poly Kernel

In [41]:

```
svm_mod = SVC(
  C = 1.0,
  kernel = 'poly',
  degree = 3,
  gamma = 'scale', # 1 / (n_features * X.var())
  probability = True,
  random_state = 425
  )
```

In [42]:

```
pipe = Pipeline(steps = [
  ("col_tf", col_tf),
  ("model", svm_mod)
  ])
pipe
```

Out[42]:

```
Pipeline(steps=[('col_tf',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_impute',
                                                                   SimpleImputer())]),
                                                  ['CompPrice', 'Income',
                                                   'Advertising', 'Population',
                                                   'Price', 'Age',
                                                   'Education']),
                                                 ('cat',
                                                  Pipeline(steps=[('cat_impute',
                                                                   SimpleImputer(strategy='most_fr
equent')),
                                                                  ('encoder',
                                                                   OneHotEncoder())]),
                                                  ['Urban', 'US',
                                                   'ShelveLoc'])])),
                ('model',
                 SVC(kernel='poly', probability=True, random_state=425))])
```

In [43]:

```
C_grid = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0]
d_grid = [2, 3, 4, 5]
tuned_parameters = {
  "model__C": C_grid,
  "model__degree": d_grid
  }
tuned_parameters
```

Out[43]:

```
{'model__C': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0], 'model__degree': [2, 3, 4, 5]}
```

In [44]:

```
n_folds = 5
search = GridSearchCV(
  pipe,
  tuned_parameters,
  cv = n_folds,
  scoring = "roc_auc",
  # Refit the best model on the whole data set
  refit = True
  )
```

In [45]:

```
search.fit(X1_train,Y1_train)

```

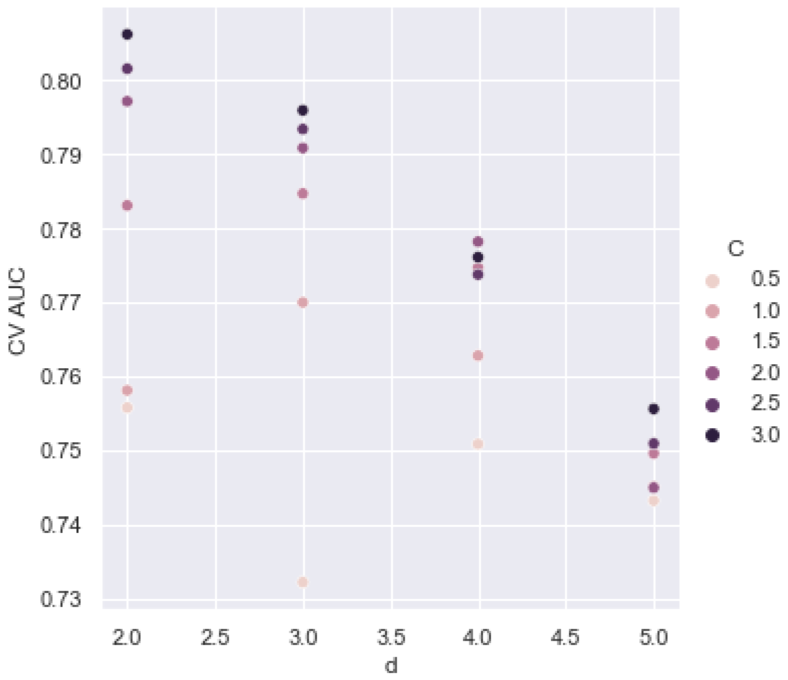
Out[45]:

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('col_tf',
                                        ColumnTransformer(transformers=[('num',
                                                                         Pipeline(steps=[('num_imp
ute',
                                                                                          SimpleIm
puter())]),
                                                                         ['CompPrice',
                                                                          'Income',
                                                                          'Advertising',
                                                                          'Population',
                                                                          'Price',
                                                                          'Age',
                                                                          'Education']),
                                                                        ('cat',
                                                                         Pipeline(steps=[('cat_imp
ute',
                                                                                          SimpleIm
puter(strategy='most_frequent')),
                                                                                         ('encode
r',
                                                                                          OneHotEn
coder())]),
                                                                         ['Urban',
                                                                          'US',
                                                                          'ShelveLoc'])])),
                                       ('model',
                                        SVC(kernel='poly', probability=True,
                                            random_state=425))]),
             param_grid={'model__C': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0],
                         'model__degree': [2, 3, 4, 5]},
             scoring='roc_auc')
```

In [46]:

```
cv_res = pd.DataFrame({
  "C": np.array(search.cv_results_["param_model__C"]),
  "auc": search.cv_results_["mean_test_score"],
  "d": search.cv_results_["param_model__degree"]
  })

plt.figure()
sns.relplot(
  # kind = "line",
  data = cv_res,
  x = "d",
  y = "auc",
  hue = "C"
  ).set(
    xlabel = "d",
    ylabel = "CV AUC"
    );
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [47]:

```
search.best_score_

```

Out[47]:

```
0.8061800931366149
```

In [48]:

```
accuracy_score(
  Y1_train,
  search.best_estimator_.predict(X1_train)
  )
```

Out[48]:

```
0.7233333333333334
```

In [49]:

```
search.best_estimator_

```

Out[49]:

```
Pipeline(steps=[('col_tf',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('num_impute',
                                                                   SimpleImputer())]),
                                                  ['CompPrice', 'Income',
                                                   'Advertising', 'Population',
                                                   'Price', 'Age',
                                                   'Education']),
                                                 ('cat',
                                                  Pipeline(steps=[('cat_impute',
                                                                   SimpleImputer(strategy='most_fr
equent')),
                                                                  ('encoder',
                                                                   OneHotEncoder())]),
                                                  ['Urban', 'US',
                                                   'ShelveLoc'])])),
                ('model',
                 SVC(C=3.0, degree=2, kernel='poly', probability=True,
                     random_state=425))])
```

In [50]:

```
roc_auc_score(
  Y1_test,
  search.best_estimator_.predict_proba(X1_test)[:, 1]
  )
```

Out[50]:

```
0.7460859092733841
```

In [51]:

```
accuracy_score(
  Y1_test,
  search.best_estimator_.predict(X1_test)
  )
```

Out[51]:

```
0.65
```

The linear kernel achieved the best test scores with an accuracy of 84%, however it was not able to outperform the boosting classification from HW4 which achieved a test score of 86%.

Comparing all models from HW4 and HW5, their accuracy performances ranks as follows:

1) Boosting 86%,

2) SVM(Linear) 84%,

3) Random Forest 83%,

4) DecisionTree 77%,

5) SVM(Poly) 65%,

6) SVM(radial) 55%

The models also ranked the same with roc_auc_score.

BONUS

We can use the formula for the signed distance from a point x to a hyperplane with normal vector w and intercept b: d = (w^T x + b) / ||w||

First, we need to find the normal vector of the hyperplane f(X) = 0. This can be done by taking the gradient of f(X) with respect to X and setting it equal to zero:

∇f(X) = [B1, B2, ..., Bp]^T = 0

So the normal vector is simply [B1, B2, ..., Bp]^T.

Next, we can plug in this normal vector and set the intercept b to 0 into the formula for the signed distance:

d = ([B1, B2, ..., Bp]^T)^T x / ||[B1, B2, ..., Bp]||

Simplifying, we get:

d = B^T x / ||B||

which shows that f(x) is proportional to the signed distance of x to the hyperplane f(X) = 0. Specifically, if f(x) > 0, then x is on one side of the hyperplane, and if f(x) < 0, then x is on the other side. If f(x) = 0, then x is on the hyperplane itself.