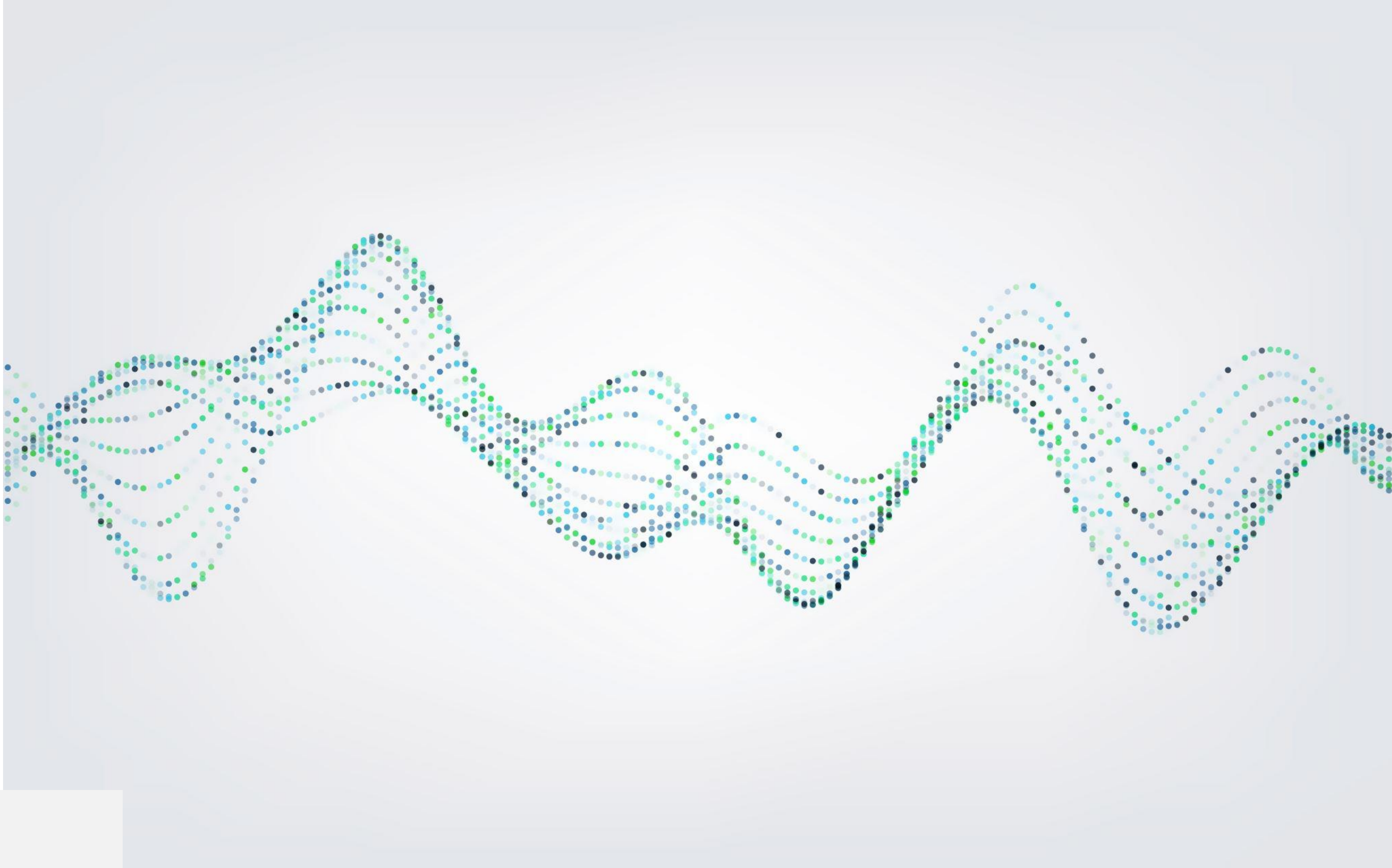
SVM

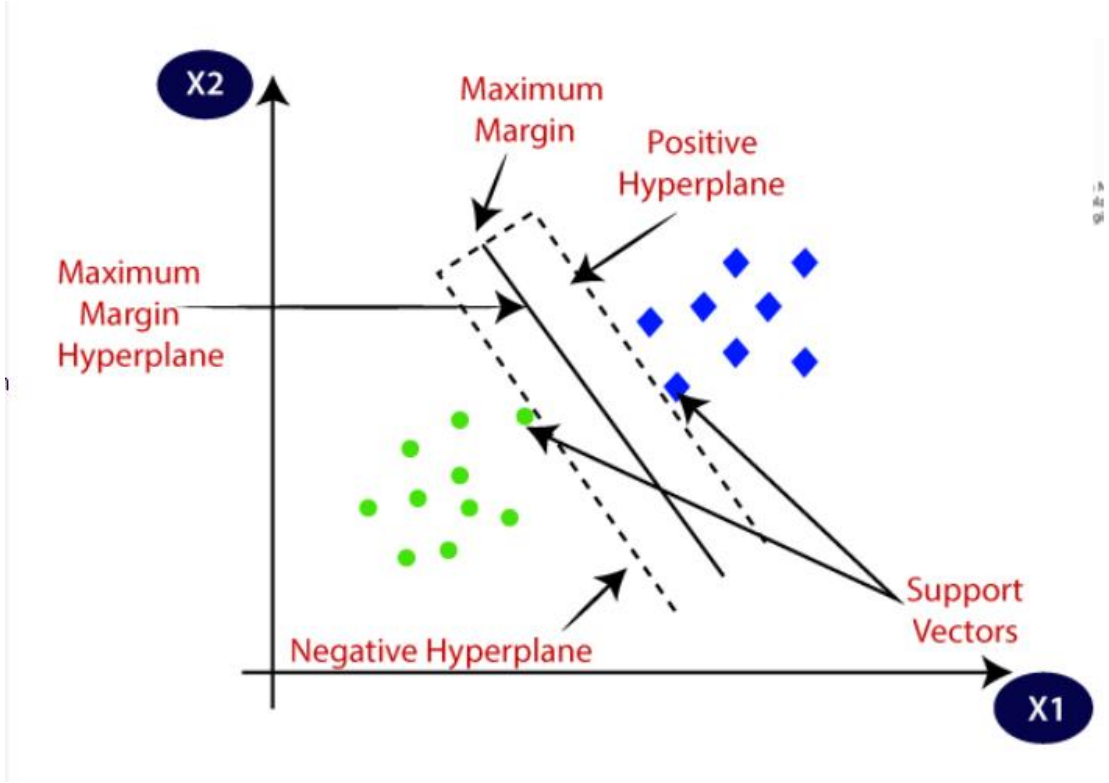


Matthieu Olislaegers

Sylvia Smolders

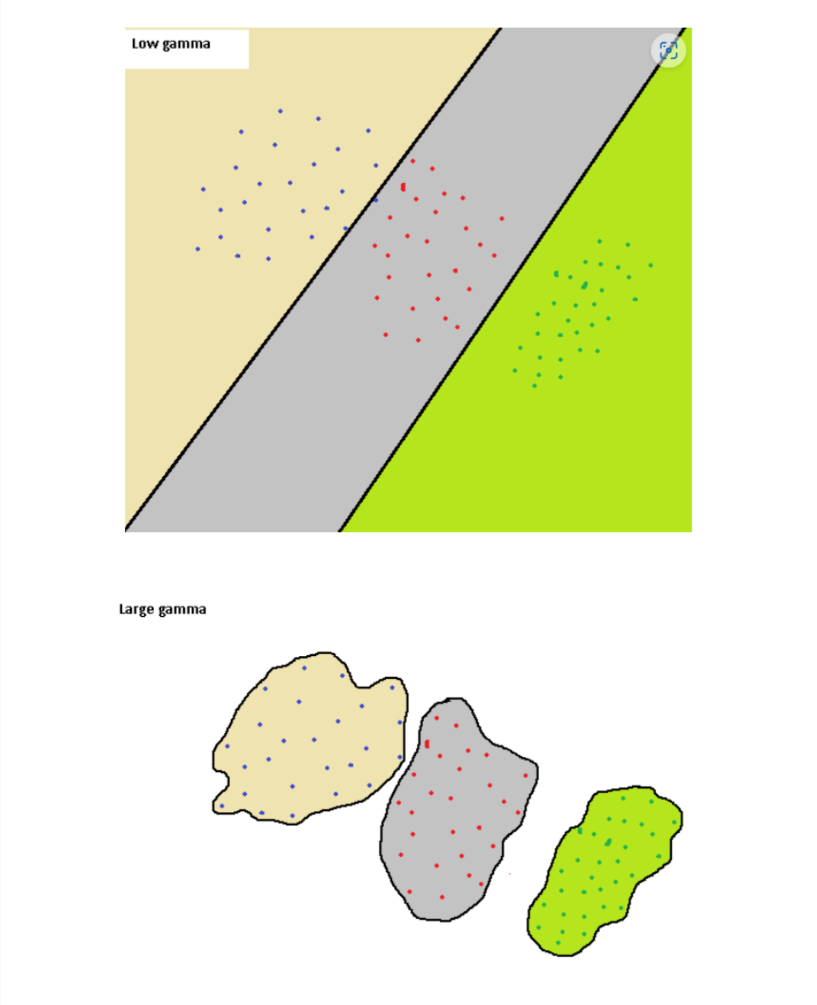
Wat is SVM

- Support Vector Machine
- Supervised Learning
- Classificatie & regressie
- Doel: beste lijn / beslissingsgrens dat n-dimensionaale ruimte kan splitsen in klassen, voor toekomstige data.
- Beslissingsgrens = hyperplane



hyperparameters

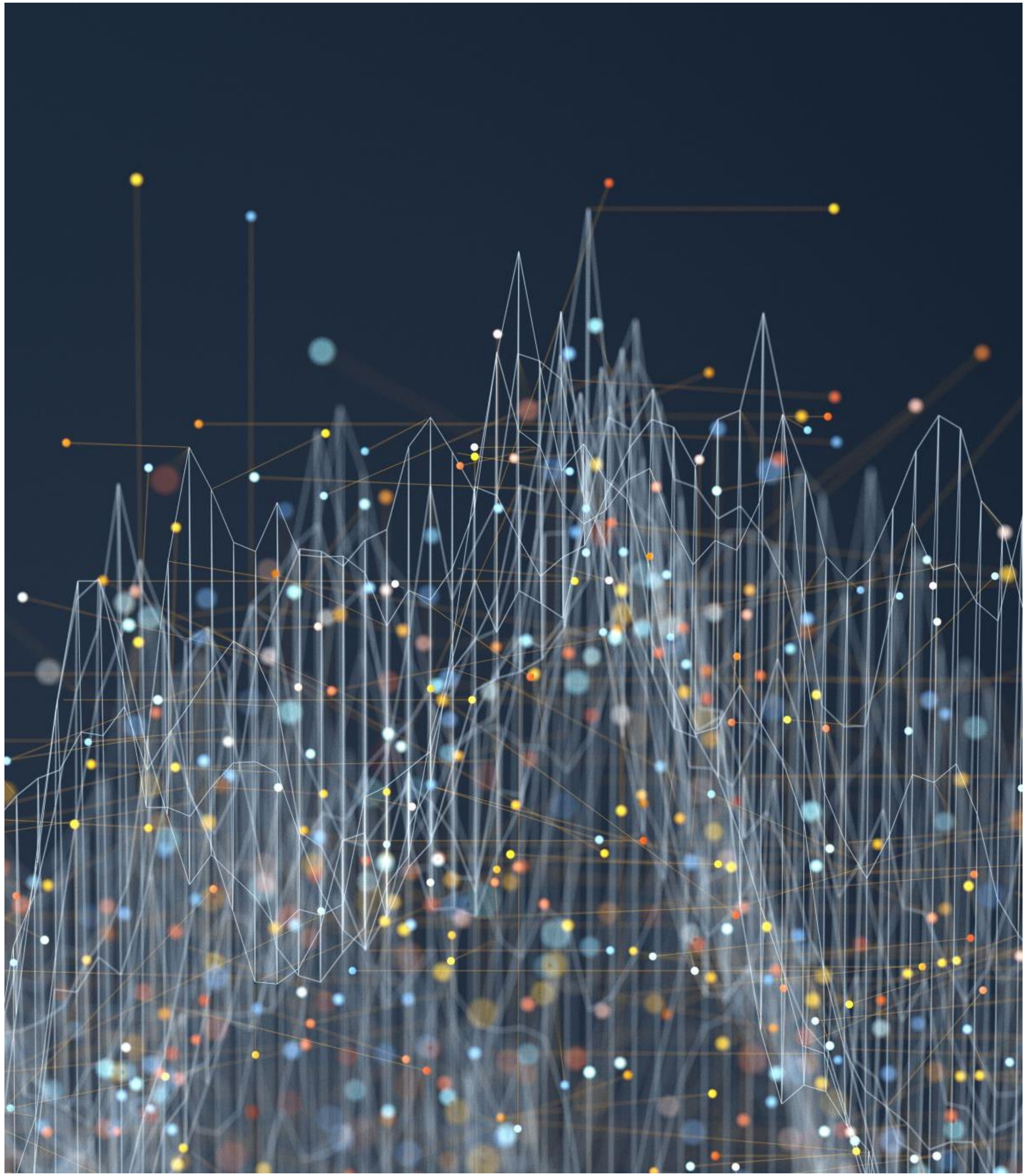
- C parameter: voor misplaatse data punt
- Kernel trick: gamma parameter of RBF, controleert de invloeide afstand van één trainingspunt



Gebruik

- Aantal features is groot t.o.v. de datapoints in de dataset
- Hoog aantal Nan waardes

- Niet geschikt voor grote datasets
- Niet voor onevenwichtige datasets
- SVMs met de verkeerde kernel
- Te veel ruis in de data



Overfitting

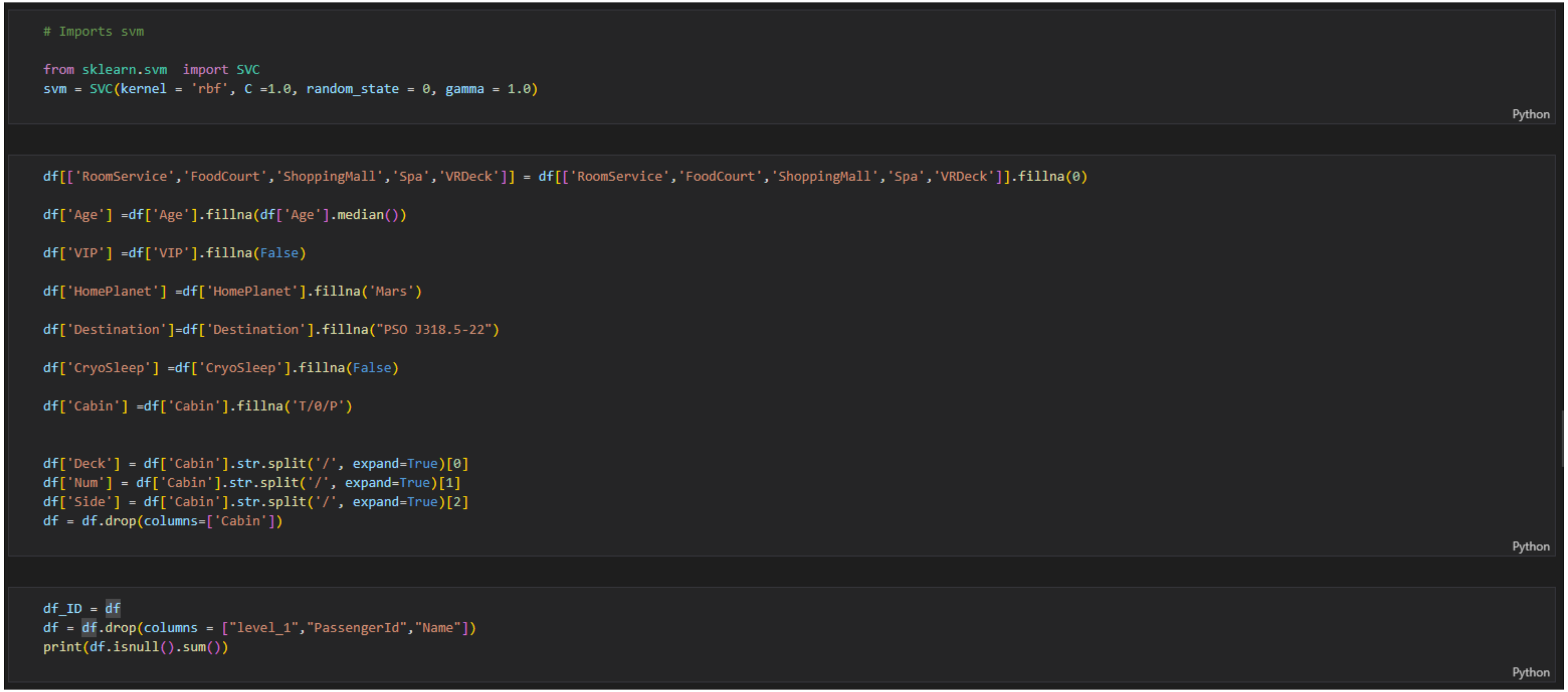
Oorzaken:

- complexe data
- traningsdata vanbuiten leren

Oplossingen:

- Cross validation
- Voor kleine datasets: splitsen van de data

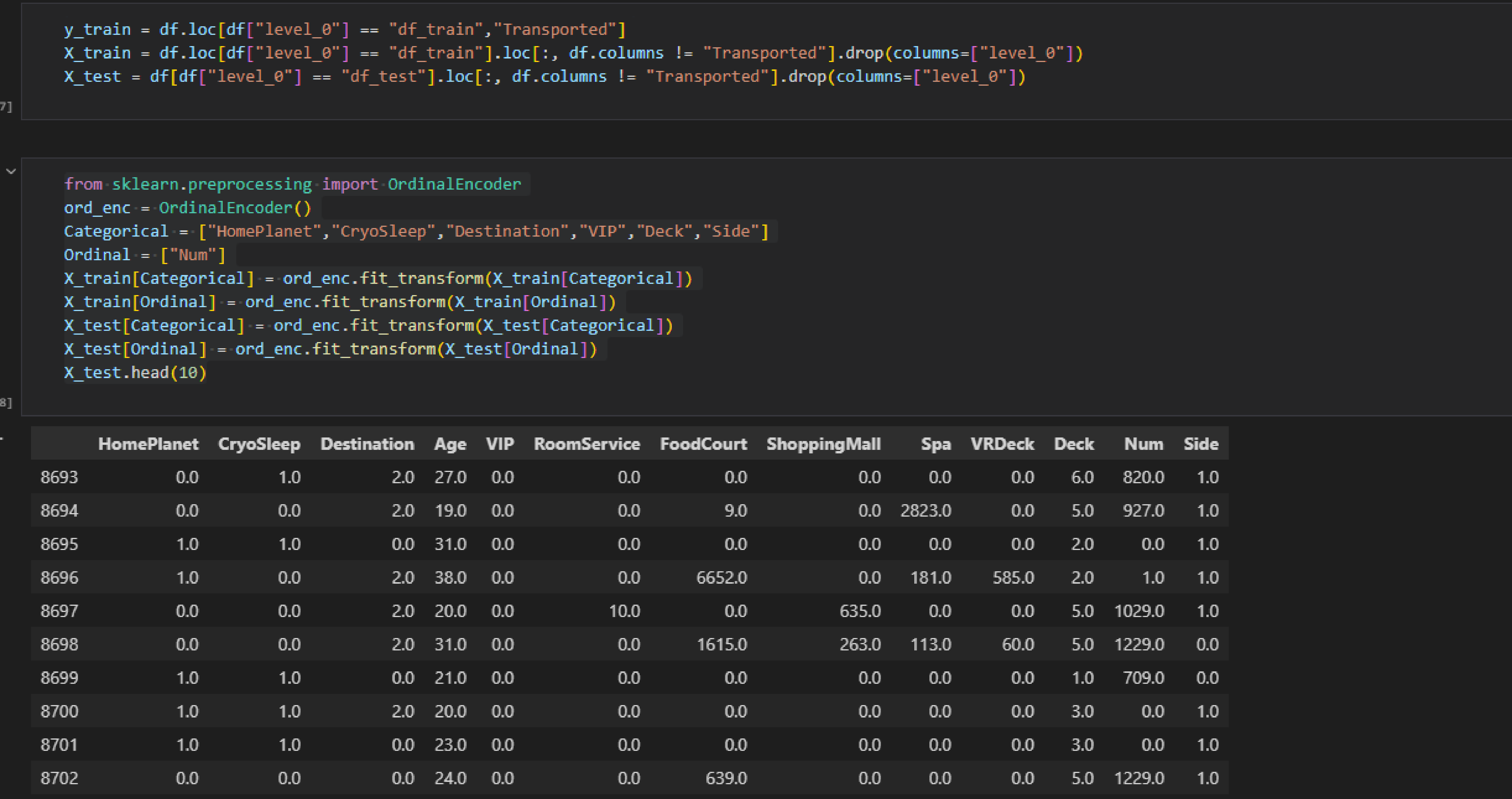
Toepassing



```
# Imports svm

from sklearn.svm  import SVC
svm = SVC(kernel = 'rbf', C =1.0, random_state = 0, gamma = 1.0)
```

```
df[['RoomService','FoodCourt','ShoppingMall','Spa','VRDeck']] = df[['RoomService','FoodCourt','ShoppingMall','Spa','VRDeck']].fillna(0)

df['Age'] =df['Age'].fillna(df['Age'].median())

df['VIP'] =df['VIP'].fillna(False)

df['HomePlanet'] =df['HomePlanet'].fillna('Mars')

df['Destination']=df['Destination'].fillna("PSO J318.5-22")

df['CryoSleep'] =df['CryoSleep'].fillna(False)

df['Cabin'] =df['Cabin'].fillna('T/0/P')


df['Deck'] = df['Cabin'].str.split('/', expand=True)[0]
df['Num'] = df['Cabin'].str.split('/', expand=True)[1]
df['Side'] = df['Cabin'].str.split('/', expand=True)[2]
df = df.drop(columns=['Cabin'])
```

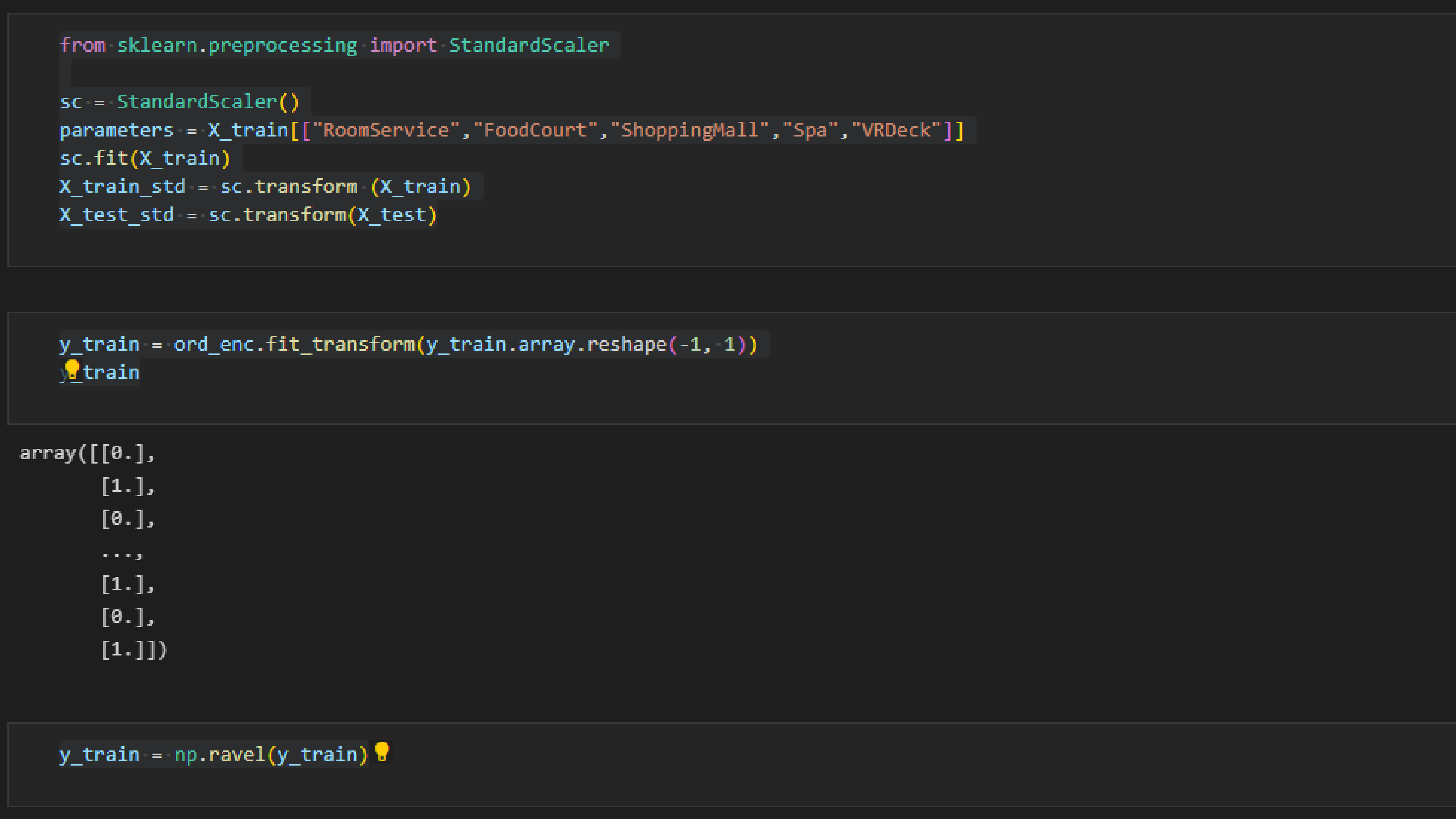
```
df_ID = df
df = df.drop(columns = ["level_1","PassengerId","Name"])
print(df.isnull().sum())
```



```python
y_train = df.loc[df["level_0"] == "df_train","Transported"]
X_train = df.loc[df["level_0"] == "df_train"].loc[:, df.columns != "Transported"].drop(columns=["level_0"])
X_test = df[df["level_0"] == "df_test"].loc[:, df.columns != "Transported"].drop(columns=["level_0"])
```

```python
from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()
Categorical = ["HomePlanet","CryoSleep","Destination","VIP","Deck","Side"]
Ordinal = ["Num"]
X_train[Categorical] = ord_enc.fit_transform(X_train[Categorical])
X_train[Ordinal] = ord_enc.fit_transform(X_train[Ordinal])
X_test[Categorical] = ord_enc.fit_transform(X_test[Categorical])
X_test[Ordinal] = ord_enc.fit_transform(X_test[Ordinal])
X_test.head(10)
```

	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck	Deck	Num	Side
8693	0.0	1.0	2.0	27.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	820.0	1.0
8694	0.0	0.0	2.0	19.0	0.0	0.0	9.0	0.0	2823.0	0.0	5.0	927.0	1.0
8695	1.0	1.0	0.0	31.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0
8696	1.0	0.0	2.0	38.0	0.0	0.0	6652.0	0.0	181.0	585.0	2.0	1.0	1.0
8697	0.0	0.0	2.0	20.0	0.0	10.0	0.0	635.0	0.0	0.0	5.0	1029.0	1.0
8698	0.0	0.0	2.0	31.0	0.0	0.0	1615.0	263.0	113.0	60.0	5.0	1229.0	0.0
8699	1.0	1.0	0.0	21.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	709.0	0.0
8700	1.0	1.0	2.0	20.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	1.0
8701	1.0	1.0	0.0	23.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	1.0
8702	0.0	0.0	0.0	24.0	0.0	0.0	639.0	0.0	0.0	0.0	5.0	1229.0	1.0

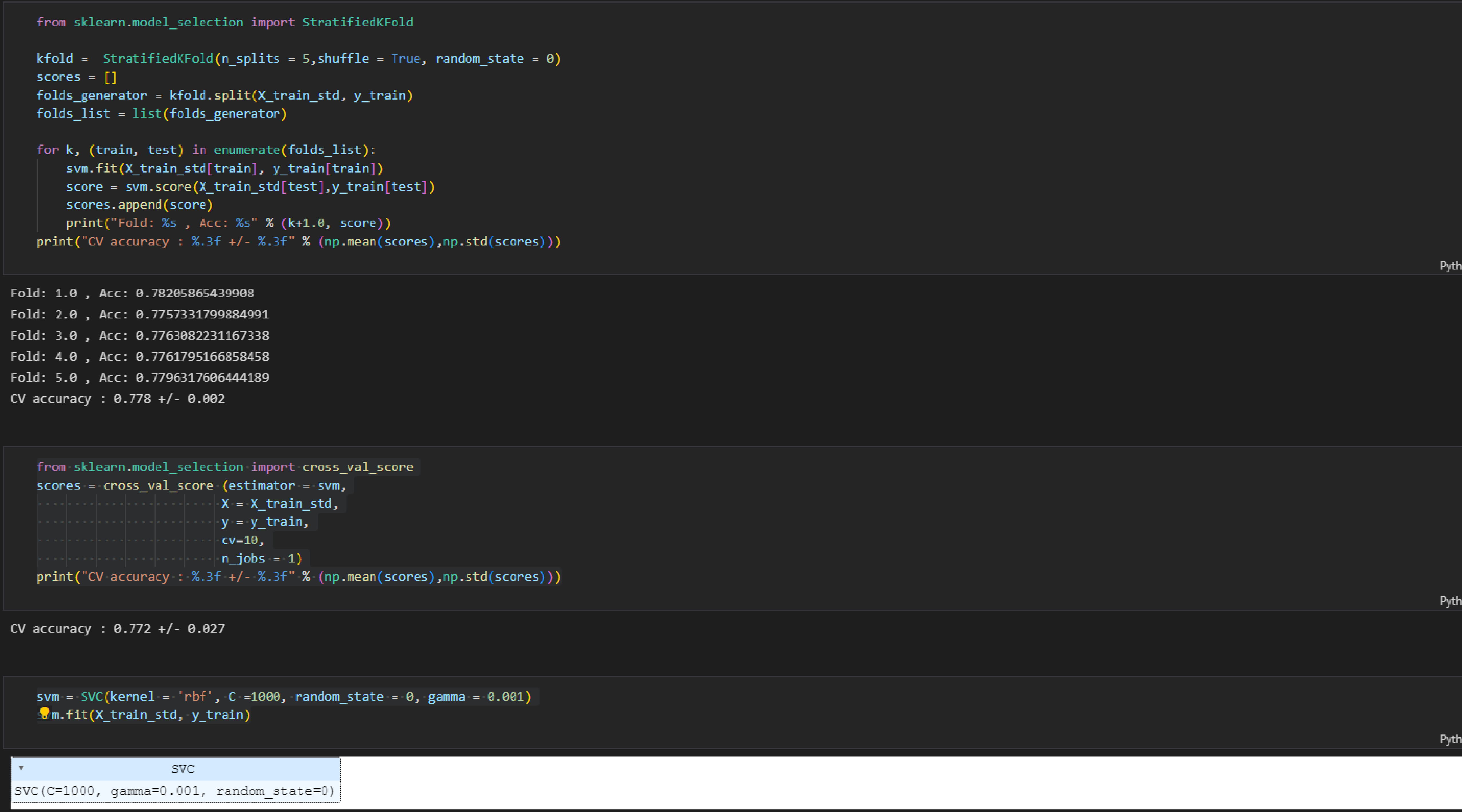


```python
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
parameters = X_train[["RoomService","FoodCourt","ShoppingMall","Spa","VRDeck"]]
sc.fit(X_train)
X_train_std = sc.transform (X_train)
X_test_std = sc.transform(X_test)
```

```python
y_train = ord_enc.fit_transform(y_train.array.reshape(-1, 1))
y_train
```

```
array([[0.],
       [1.],
       [0.],
       ...,
       [1.],
       [0.],
       [1.]])
```

```python
y_train = np.ravel(y_train)
```



```python
from sklearn.model_selection import StratifiedKFold

kfold =  StratifiedKFold(n_splits = 5,shuffle = True, random_state = 0)
scores = []
folds_generator = kfold.split(X_train_std, y_train)
folds_list = list(folds_generator)

for k, (train, test) in enumerate(folds_list):
    svm.fit(X_train_std[train], y_train[train])
    score = svm.score(X_train_std[test],y_train[test])
    scores.append(score)
    print("Fold: %s , Acc: %s" % (k+1.0, score))
print("CV accuracy : %.3f +/- %.3f" % (np.mean(scores),np.std(scores)))
```

```
Fold: 1.0 , Acc: 0.78205865439908
Fold: 2.0 , Acc: 0.7757331799884991
Fold: 3.0 , Acc: 0.7763082231167338
Fold: 4.0 , Acc: 0.7761795166858458
Fold: 5.0 , Acc: 0.7796317606444189
CV accuracy : 0.778 +/- 0.002
```

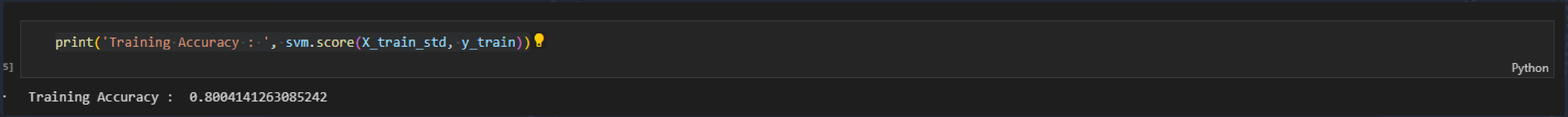
```python
from sklearn.model_selection import cross_val_score
scores = cross_val_score (estimator = svm,
                          X = X_train_std,
                          y = y_train,
                          cv=10,
                          n_jobs = 1)
print("CV accuracy : %.3f +/- %.3f" % (np.mean(scores),np.std(scores)))
```

```
CV accuracy : 0.772 +/- 0.027
```

```python
svm = SVC(kernel = 'rbf', C =1000, random_state = 0, gamma = 0.001)
svm.fit(X_train_std, y_train)
```

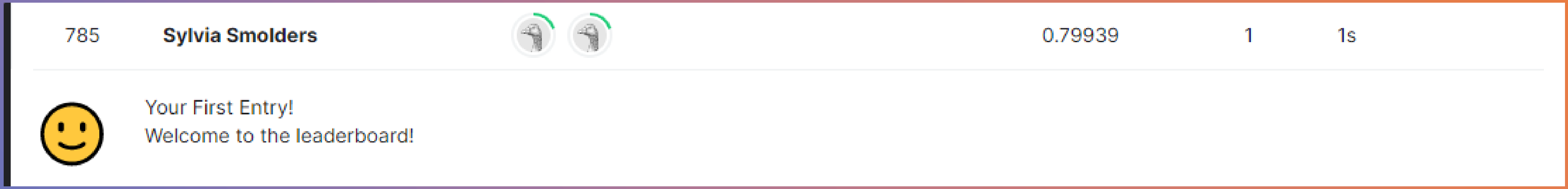
```
SVC(C=1000, gamma=0.001, random_state=0)
```

SVM score



```python
print('Training Accuracy : ', svm.score(X_train_std, y_train))
```

```
Training Accuracy :  0.8004141263085242
```



785	**Sylvia Smolders**	0.79939	1	1s

Your First Entry!
Welcome to the leaderboard!

Resultaat Kaggle