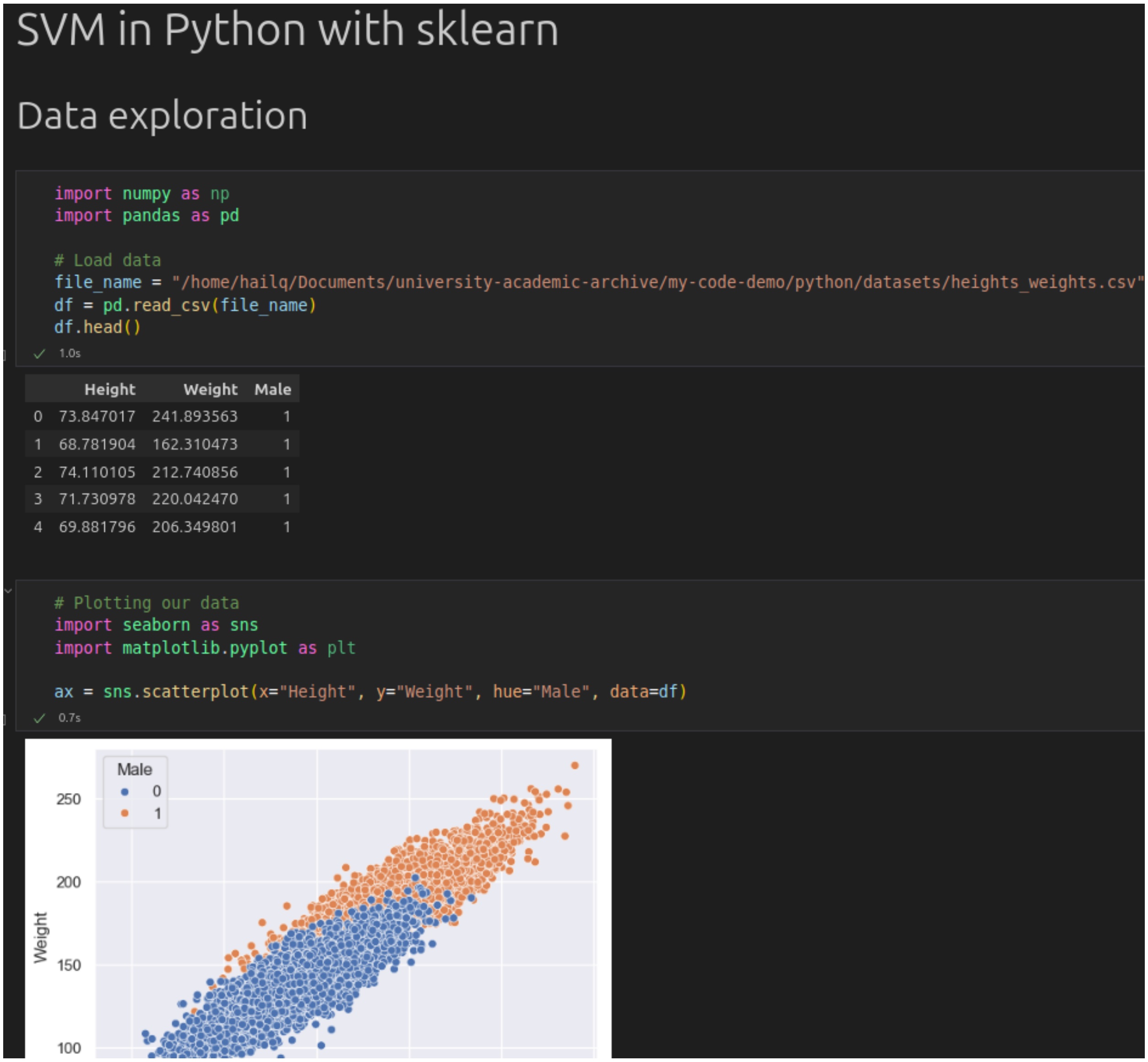
Week 4 Learning Activities

Example exploration

Basic SVM Classification



SVM in Python with sklearn

Data exploration

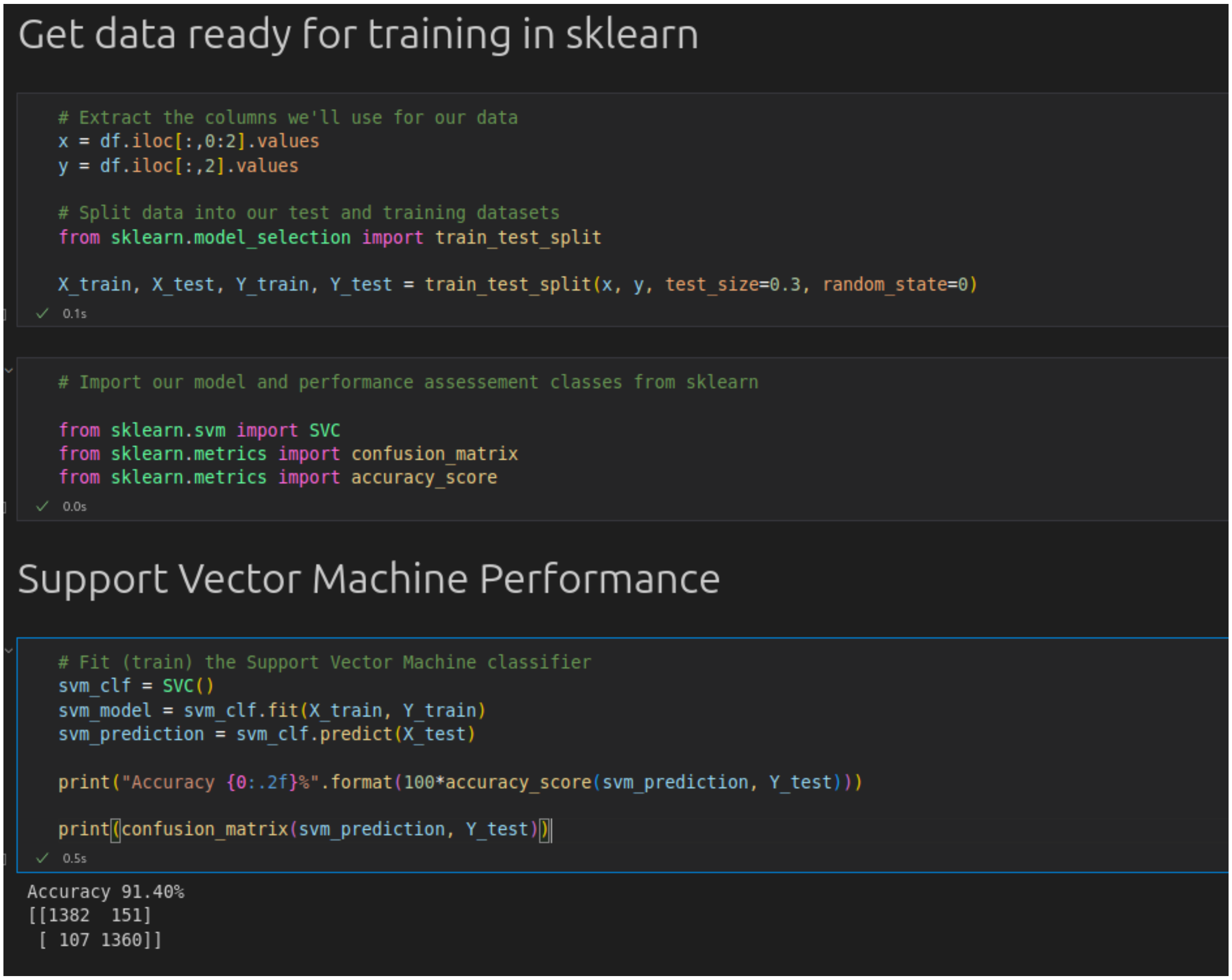
```
import numpy as np
import pandas as pd

# Load data
file_name = "/home/hailq/Documents/university-academic-archive/my-code-demo/python/datasets/heights_weights.csv"
df = pd.read_csv(file_name)
df.head()
```

✓ 1.0s

	Height	Weight	Male
0	73.847017	241.893563	1
1	68.781904	162.310473	1
2	74.110105	212.740856	1
3	71.730978	220.042470	1
4	69.881796	206.349801	1

```
# Plotting our data
import seaborn as sns
import matplotlib.pyplot as plt

ax = sns.scatterplot(x="Height", y="Weight", hue="Male", data=df)
```

✓ 0.7s



Get data ready for training in sklearn

```
# Extract the columns we'll use for our data
x = df.iloc[:,0:2].values
y = df.iloc[:,2].values

# Split data into our test and training datasets
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

✓ 0.1s

```
# Import our model and performance assessement classes from sklearn

from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

✓ 0.0s

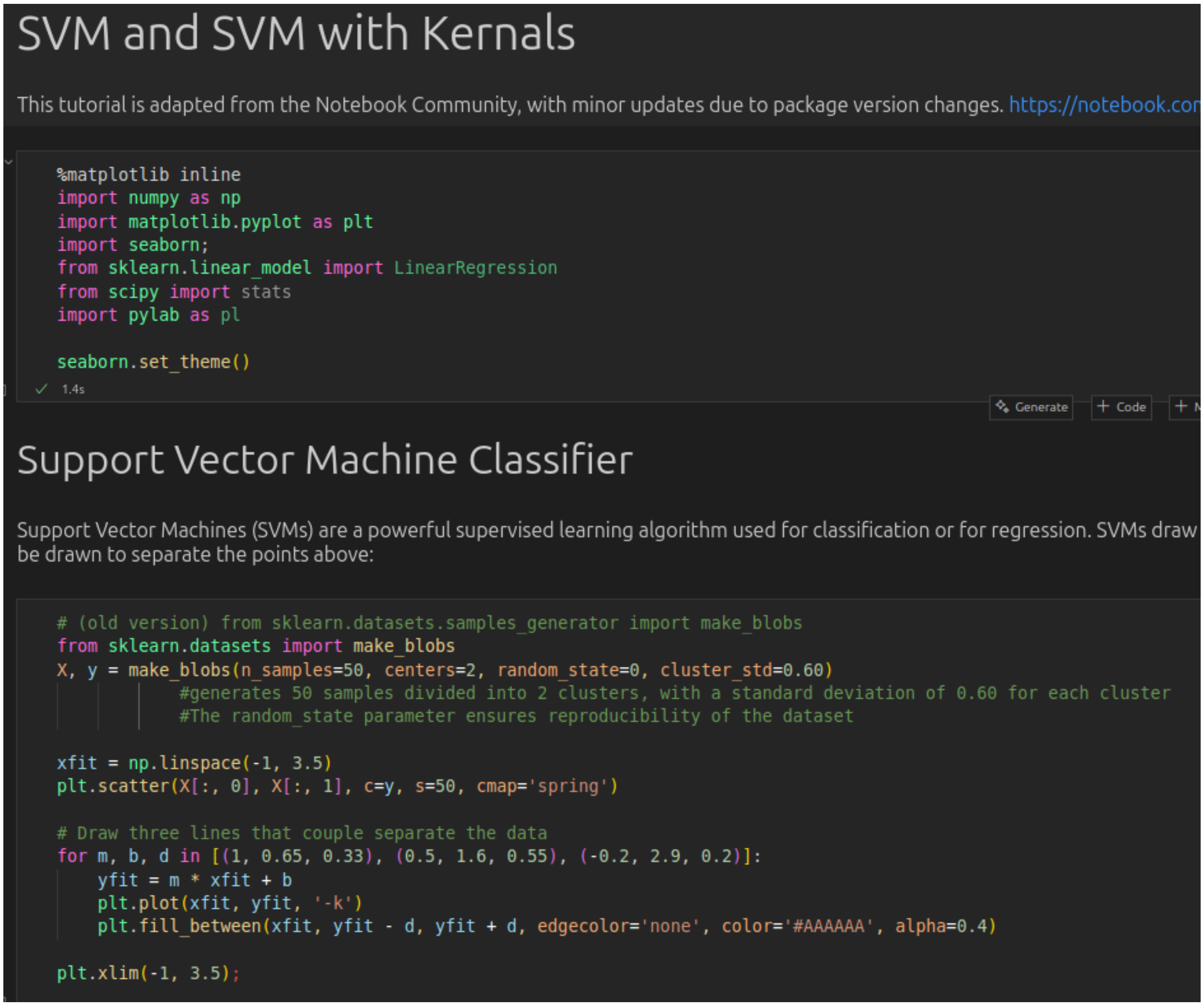
Support Vector Machine Performance

```
# Fit (train) the Support Vector Machine classifier
svm_clf = SVC()
svm_model = svm_clf.fit(X_train, Y_train)
svm_prediction = svm_clf.predict(X_test)

print("Accuracy {0:.2f}%".format(100*accuracy_score(svm_prediction, Y_test)))

print(confusion_matrix(svm_prediction, Y_test))
```

✓ 0.5s

```
Accuracy 91.40%
[[1382  151]
 [ 107 1360]]
```

SVM with Kernels



SVM and SVM with Kernals

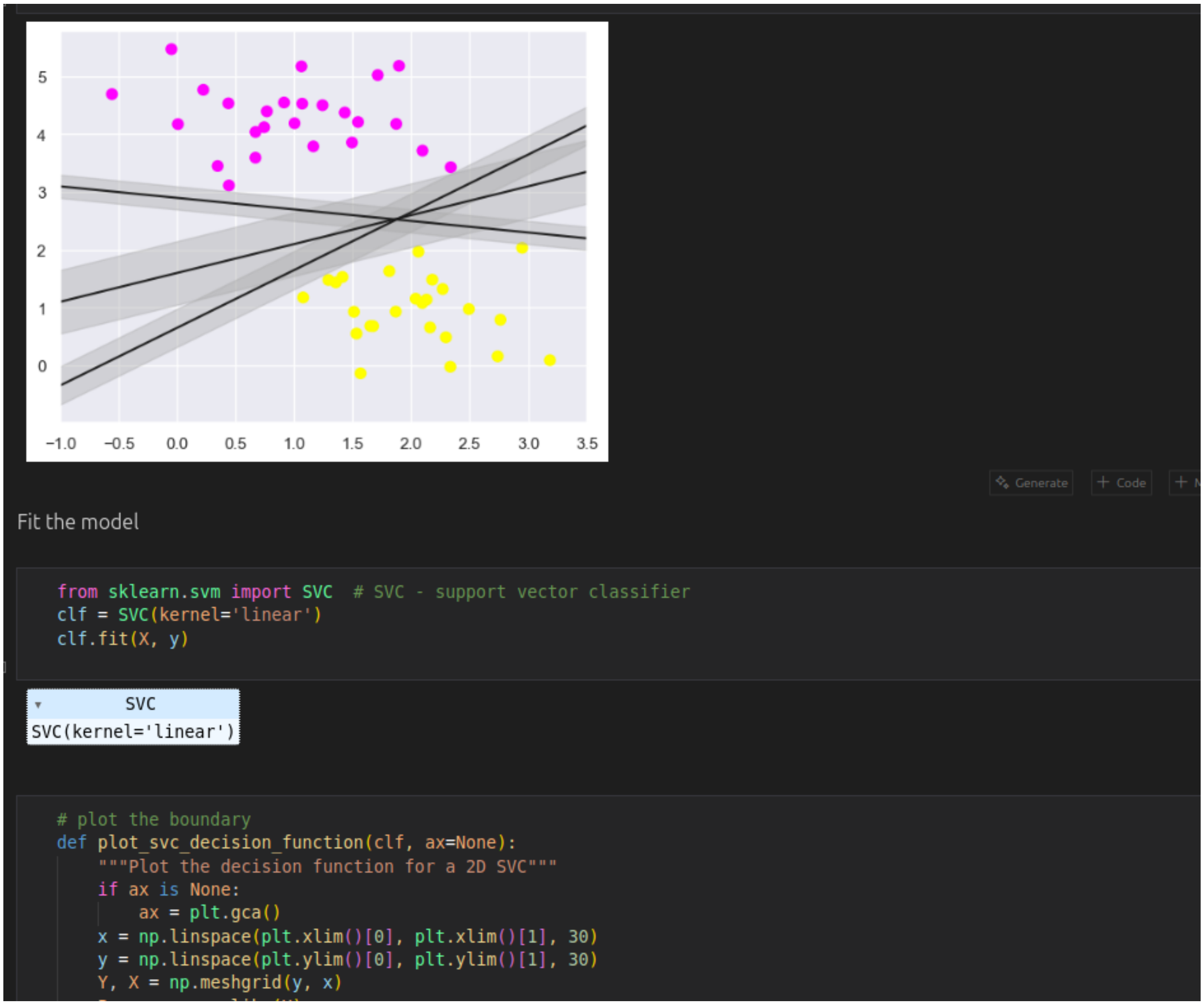
This tutorial is adapted from the Notebook Community, with minor updates due to package version changes. https://notebook.cor

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn;
from sklearn.linear_model import LinearRegression
from scipy import stats
import pylab as pl

seaborn.set_theme()
```

Support Vector Machine Classifier

Support Vector Machines (SVMs) are a powerful supervised learning algorithm used for classification or for regression. SVMs draw
be drawn to separate the points above:

```
# (old version) from sklearn.datasets.samples_generator import make_blobs
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=50, centers=2, random_state=0, cluster_std=0.60)
            #generates 50 samples divided into 2 clusters, with a standard deviation of 0.60 for each cluster
            #The random_state parameter ensures reproducibility of the dataset

xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='spring')

# Draw three lines that couple separate the data
for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none', color='#AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);
```

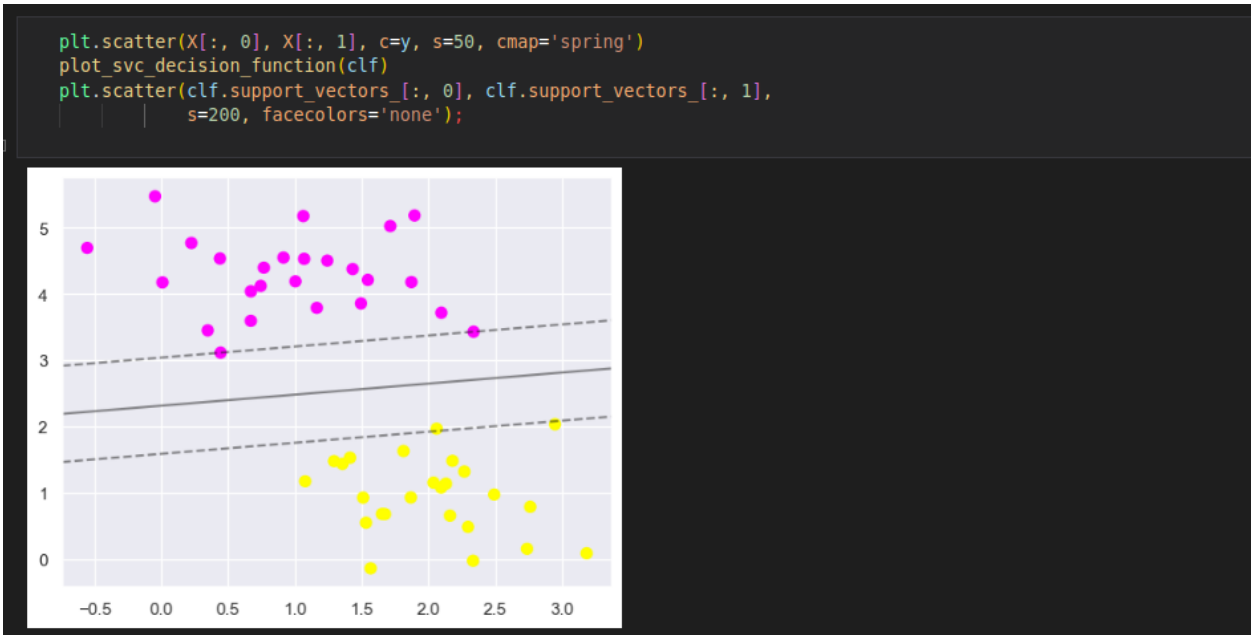


Fit the model

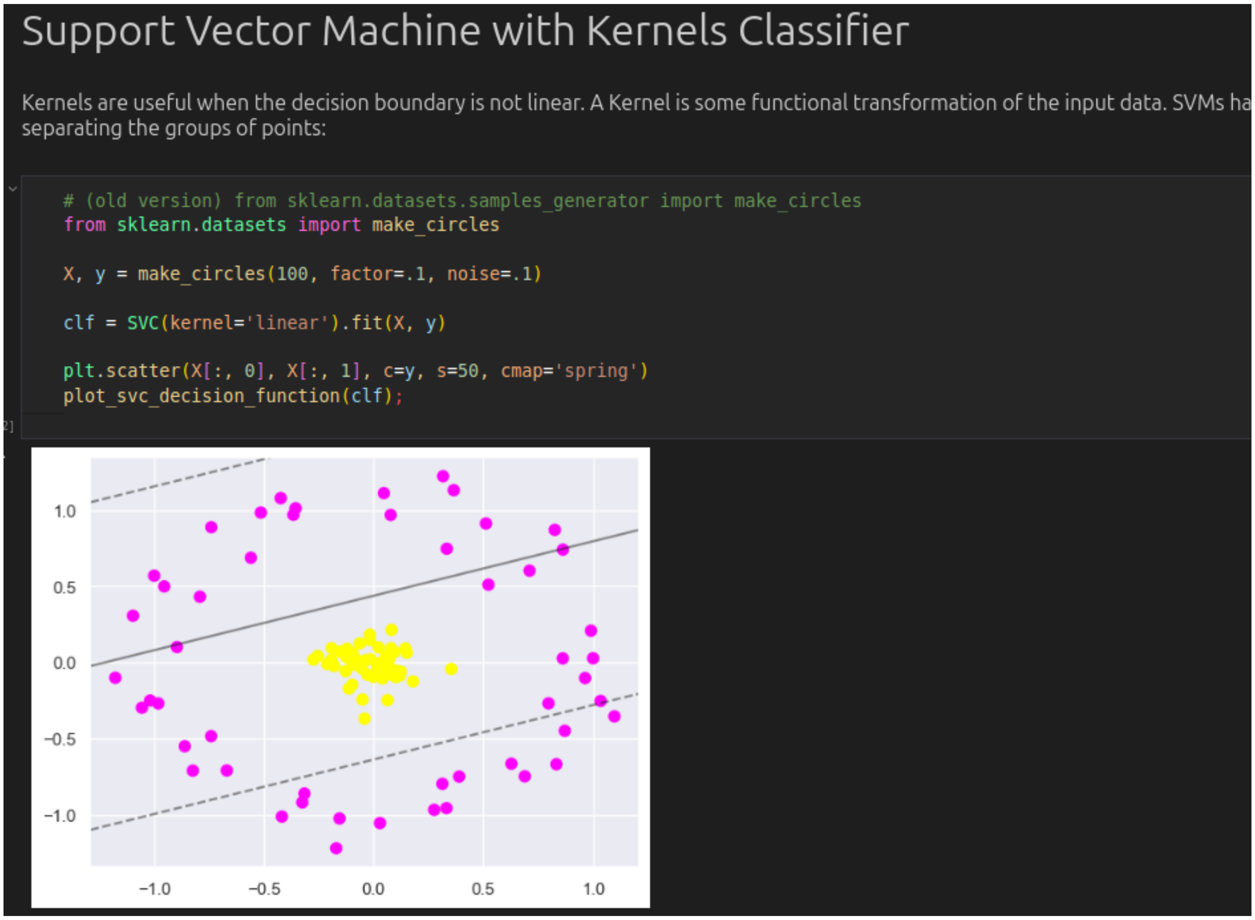
```
from sklearn.svm import SVC  # SVC - support vector classifier
clf = SVC(kernel='linear')
clf.fit(X, y)
```

```
SVC
SVC(kernel='linear')
```

```
# plot the boundary
def plot_svc_decision_function(clf, ax=None):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    x = np.linspace(plt.xlim()[0], plt.xlim()[1], 30)
    y = np.linspace(plt.ylim()[0], plt.ylim()[1], 30)
    Y, X = np.meshgrid(y, x)
```



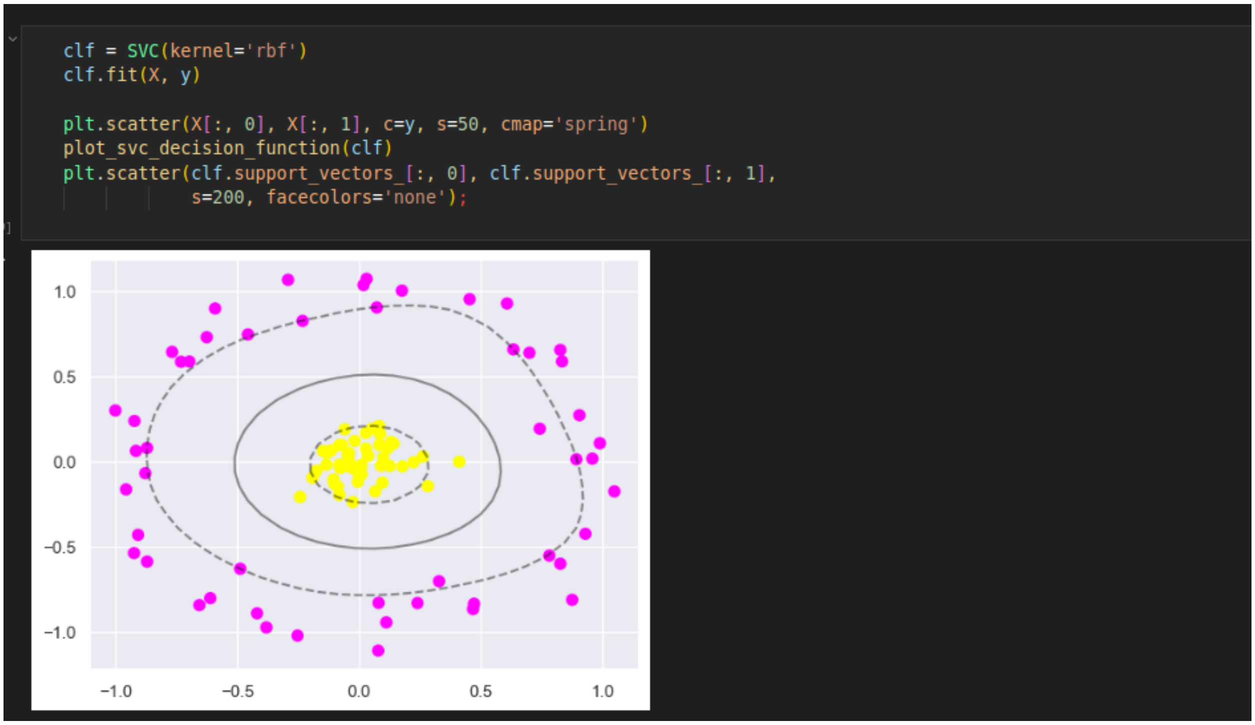
```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='spring')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
            s=200, facecolors='none');
```



Support Vector Machine with Kernels Classifier

Kernels are useful when the decision boundary is not linear. A Kernel is some functional transformation of the input data. SVMs ha
separating the groups of points:

```
# (old version) from sklearn.datasets.samples_generator import make_circles
from sklearn.datasets import make_circles

X, y = make_circles(100, factor=.1, noise=.1)

clf = SVC(kernel='linear').fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='spring')
plot_svc_decision_function(clf);
```



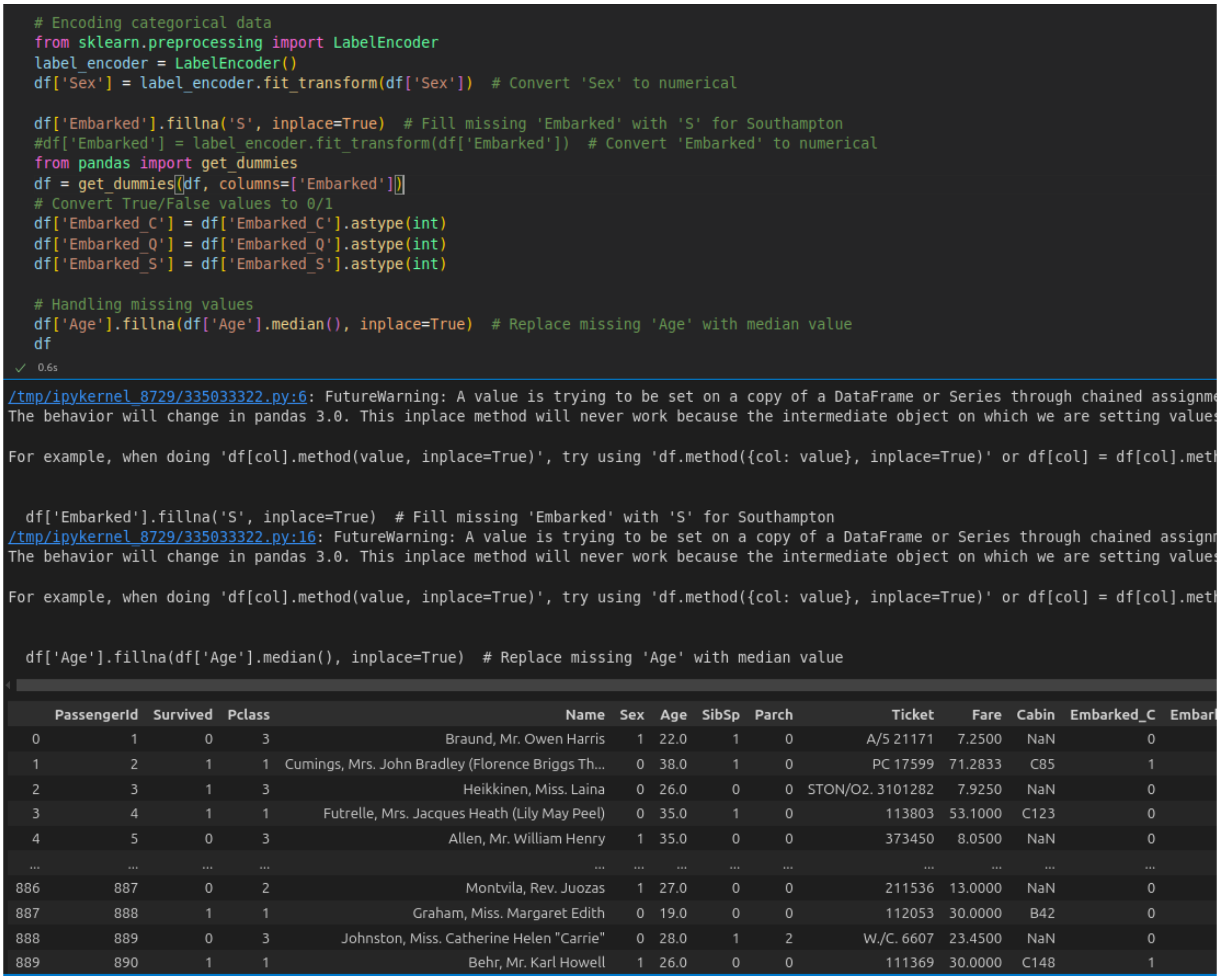
```
clf = SVC(kernel='rbf')
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='spring')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
            s=200, facecolors='none');
```

Custom SVM Classifier

Feature Selection

Since we are using the dataset from week 3 activities, all the preprocessing steps taken in the last tutorial can be re-used. Specifically:

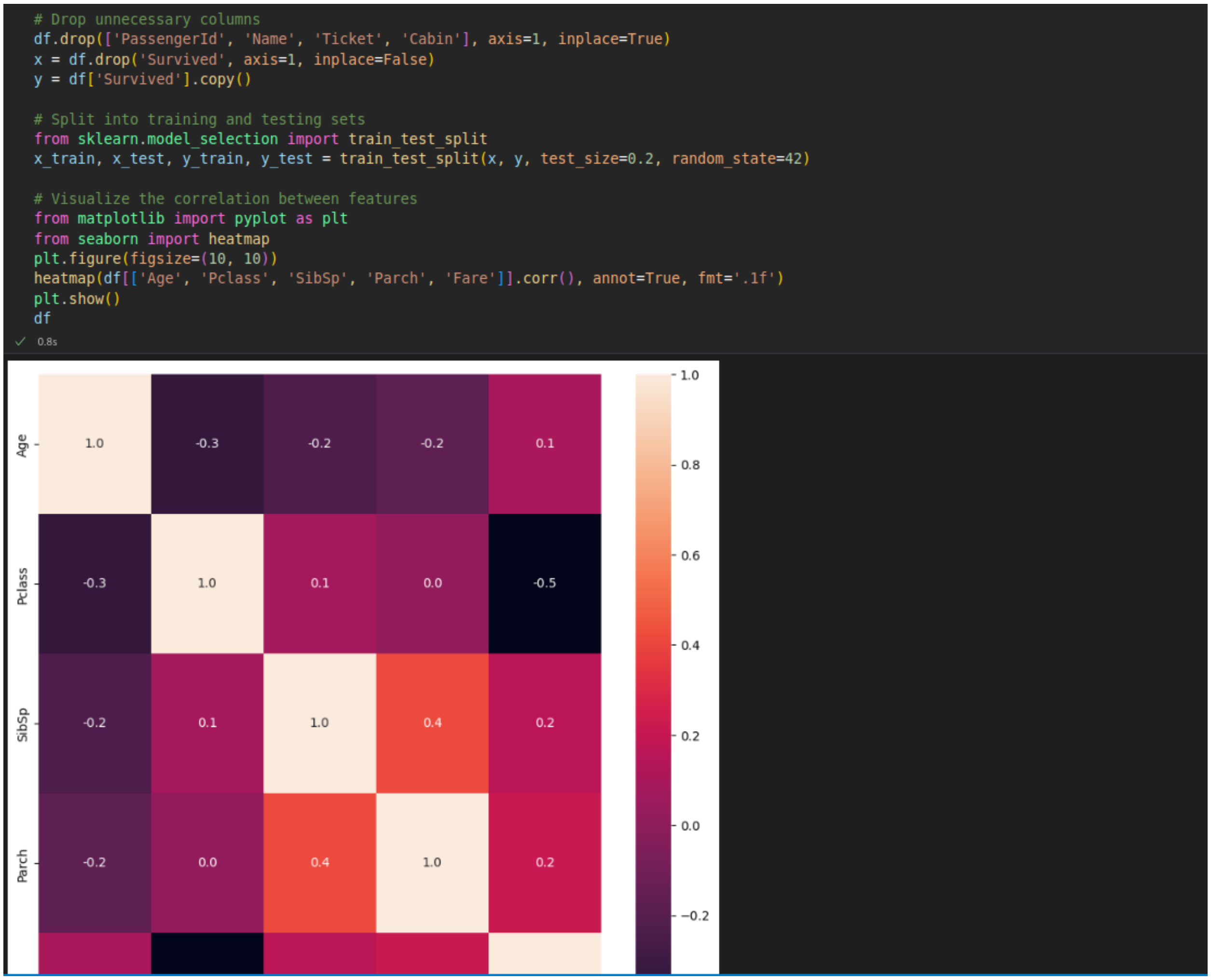
- The ‘sex’ column should be converted into boolean value (male or not male) rather than string values. In the scope of the lab session, we do not consider other recognized gender or sexual sorts.
- The ‘embarked’ attribute is transformed via one-hot encoding method, filling the empty cell with a specific representative value.
- Missing value in ‘age’ column is replaced by the median value of the whole dataset.
- The passenger id and name should not be an input attribute. Though the name contains one’s title (Mr., Ms., …) we have the ‘sex’ that bears almost the same meaning.
- The ‘ticket’ column comprises different nominal data that is almost impossible to preprocess.
- For the ‘cabin’ column, it is unclear how they are categorized (for instance, A and B for first class, …). Therefore, without clear knowledge of this we might want to ignore this feature as well.
- We use the drop() method with inplace parameter set to True from DataFrame class to remove unwanted attributes



```python
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df['Sex'])  # Convert 'Sex' to numerical

df['Embarked'].fillna('S', inplace=True)  # Fill missing 'Embarked' with 'S' for Southampton
#df['Embarked'] = label_encoder.fit_transform(df['Embarked'])  # Convert 'Embarked' to numerical
from pandas import get_dummies
df = get_dummies(df, columns=['Embarked'])
# Convert True/False values to 0/1
df['Embarked_C'] = df['Embarked_C'].astype(int)
df['Embarked_Q'] = df['Embarked_Q'].astype(int)
df['Embarked_S'] = df['Embarked_S'].astype(int)

# Handling missing values
df['Age'].fillna(df['Age'].median(), inplace=True)  # Replace missing 'Age' with median value
df
```

```
/tmp/ipykernel_8729/335033322.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].meth


  df['Embarked'].fillna('S', inplace=True)  # Fill missing 'Embarked' with 'S' for Southampton
/tmp/ipykernel_8729/335033322.py:16: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].meth


  df['Age'].fillna(df['Age'].median(), inplace=True)  # Replace missing 'Age' with median value
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked_C	Embar
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	NaN	0	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C85	1	
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	NaN	0	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	C123	0	
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	NaN	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	1	27.0	0	0	211536	13.0000	NaN	0	
887	888	1	1	Graham, Miss. Margaret Edith	0	19.0	0	0	112053	30.0000	B42	0	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	0	28.0	1	2	W./C. 6607	23.4500	NaN	0	
889	890	1	1	Behr, Mr. Karl Howell	1	26.0	0	0	111369	30.0000	C148	1	

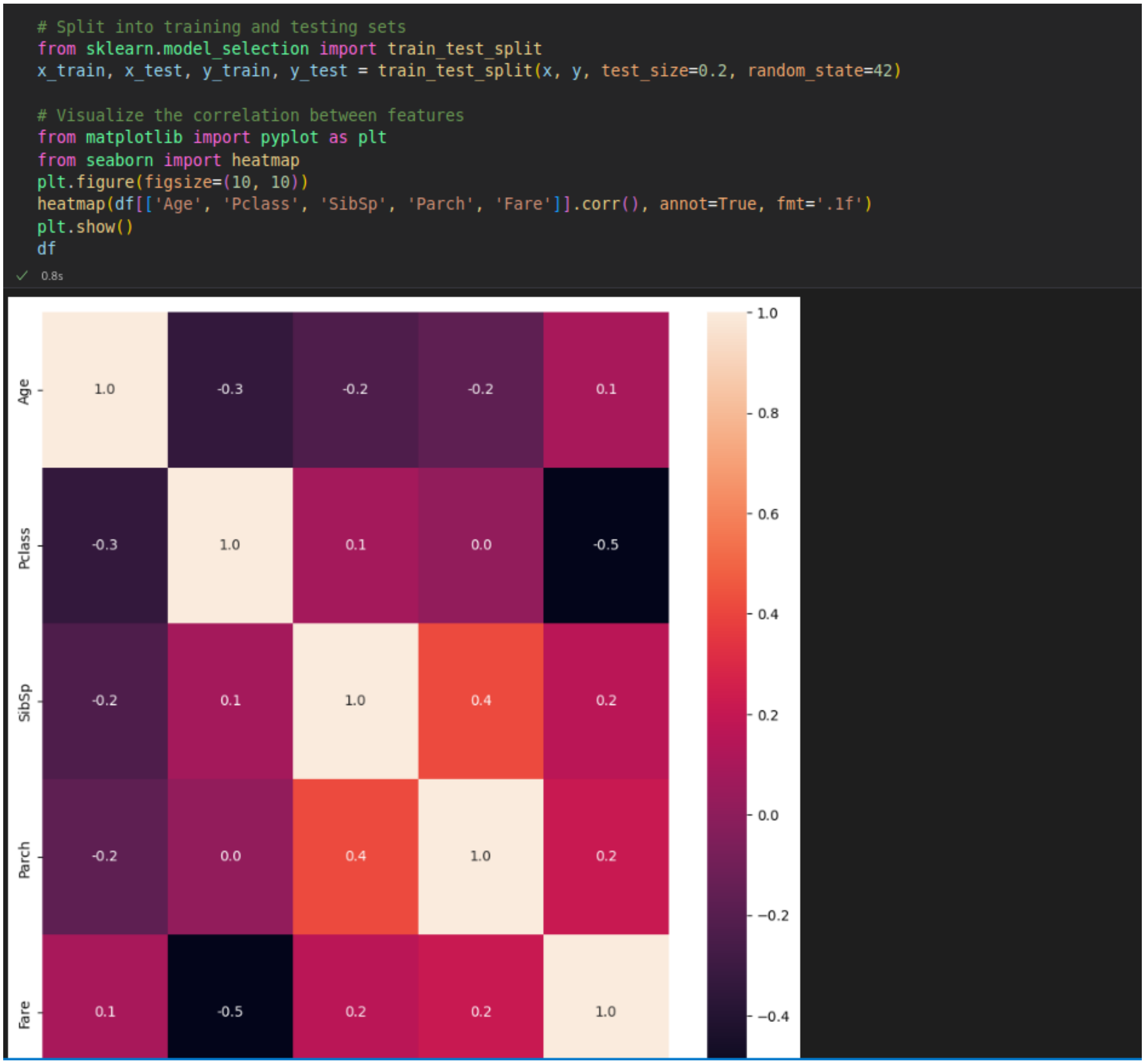


```
# Drop unnecessary columns
df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
x = df.drop('Survived', axis=1, inplace=False)
y = df['Survived'].copy()

# Split into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Visualize the correlation between features
from matplotlib import pyplot as plt
from seaborn import heatmap
plt.figure(figsize=(10, 10))
heatmap(df[['Age', 'Pclass', 'SibSp', 'Parch', 'Fare']].corr(), annot=True, fmt='.1f')
plt.show()
df
```

Data partition

The target variable should whether the person survived or not, therefore is the ‘survived’ column

We split the data into input variables (x) and output variables (y) then split the data into training set and test set utilizing scikit-learn and pandas built-in methods.



```
# Split into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Visualize the correlation between features
from matplotlib import pyplot as plt
from seaborn import heatmap
plt.figure(figsize=(10, 10))
heatmap(df[['Age', 'Pclass', 'SibSp', 'Parch', 'Fare']].corr(), annot=True, fmt='.1f')
plt.show()
df
```

✓ 0.8s

Age	1.0	-0.3	-0.2	-0.2	0.1
Pclass	-0.3	1.0	0.1	0.0	-0.5
SibSp	-0.2	0.1	1.0	0.4	0.2
Parch	-0.2	0.0	0.4	1.0	0.2
Fare	0.1	-0.5	0.2	0.2	1.0

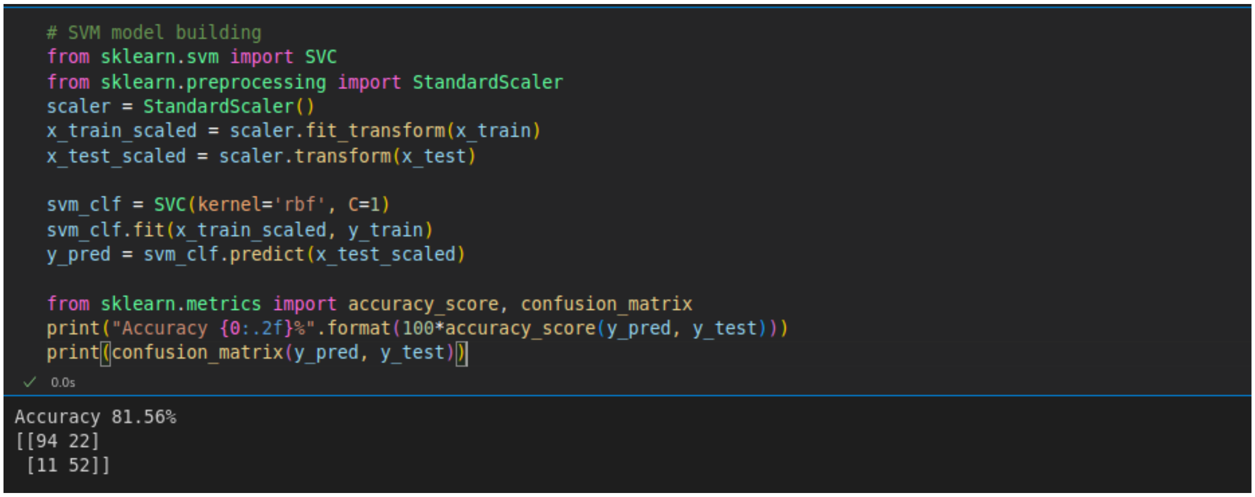
Model Building and Evaluation

In this experiment, a Support Vector Machine (SVM) classifier was built using the Radial Basis Function (RBF) kernel, a popular choice for non-linear classification tasks. The goal was to scale the dataset, train the SVM model, make predictions, and evaluate its performance based on accuracy and confusion matrix.

Before applying the SVM model, the data needed to be standardized. Standardization ensures that all features contribute equally to the model by transforming the dataset such that each feature has a mean of 0 and a standard deviation of 1.

After training the model, predictions were made on the scaled test dataset using the predict method:

The accuracy score offers a quick evaluation of overall performance, while the confusion matrix provides a deeper understanding of the model's ability to distinguish between different classes.



```
# SVM model building
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

svm_clf = SVC(kernel='rbf', C=1)
svm_clf.fit(x_train_scaled, y_train)
y_pred = svm_clf.predict(x_test_scaled)

from sklearn.metrics import accuracy_score, confusion_matrix
print("Accuracy {0:.2f}%".format(100*accuracy_score(y_pred, y_test)))
print(confusion_matrix(y_pred, y_test))
```

```
Accuracy 81.56%
[[94 22]
 [11 52]]
```