```
%pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.12/dist-packages (1.7.4.5)
Requirement already satisfied: bleach in /usr/local/lib/python3.12/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2025.8.3
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/dist-packages (from kaggle) (3.4.3)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages (from kaggle) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.9.
Requirement already satisfied: python-slugify in /usr/local/lib/python3.12/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.32.4)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.12/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.12/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.12/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.5.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packages (from kaggle) (0.5.1)
```

Download dataset

```
import kagglehub
import pandas as pd
path = kagglehub.dataset_download("henriqueyamahata/bank-marketing")

print("Path to dataset files:", path)
print(path)
data = pd.read_csv(path + "/bank-additional-full.csv", sep=";")
data.head()
```

```
Using Colab cache for faster access to the 'bank-marketing' dataset.
Path to dataset files: /kaggle/input/bank-marketing
/kaggle/input/bank-marketing
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previo
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	

5 rows × 21 columns

Dataset Preprocessing

```
data["y"] = data["y"].map({"yes": 1, "no": 0})  # convert target to binary
```

```
categorical_cols = data.select_dtypes(include=["object"]).columns.tolist()
numerical_cols = data.select_dtypes(include=["int64", "float64"]).columns.tolist()

print("Categorical:", categorical_cols)
print("Numerical:", numerical_cols)
```

```
Categorical: ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'pout
Numerical: ['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx',
```

```
df = pd.get_dummies(data, columns=categorical_cols, drop_first=True)
```

```
df.head()
```

n	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...
1	999	0	1.1	93.994	-36.4	4.857	5191.0	...
1	999	0	1.1	93.994	-36.4	4.857	5191.0	...
1	999	0	1.1	93.994	-36.4	4.857	5191.0	...
1	999	0	1.1	93.994	-36.4	4.857	5191.0	...
1	999	0	1.1	93.994	-36.4	4.857	5191.0	...

Model Training

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
X = df.drop("y", axis=1)
y = df["y"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

svm = LinearSVC(random_state=42, max_iter=5000)
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)
```

Inference

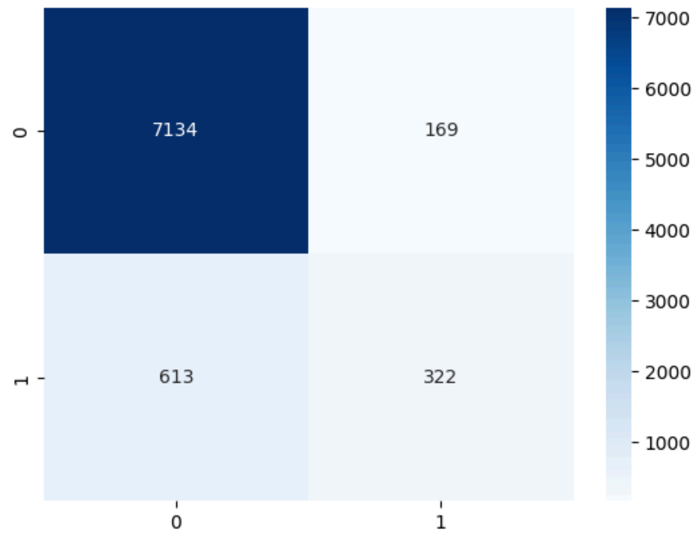
```
y_pred = svm.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)
confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", confusion_mat)
```

```
Accuracy: 0.9050740470988103
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.98      0.95      7303
           1       0.66      0.34      0.45       935

    accuracy                           0.91      8238
   macro avg       0.79      0.66      0.70      8238
weighted avg       0.89      0.91      0.89      8238

Confusion Matrix:
 [[7134  169]
 [ 613  322]]
```

```
import seaborn as sns
sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Blues")
```

```
<Axes: >
```



Non-Linear kernel SVM

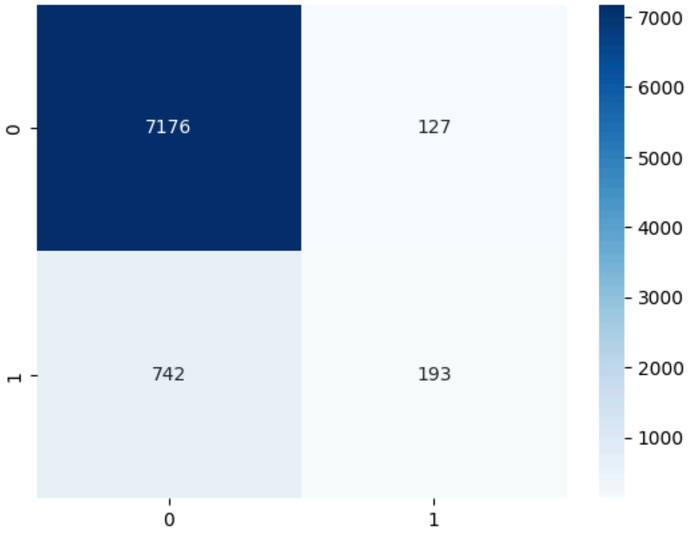
```
svm_rbf = SVC(kernel="rbf", C=1.0, gamma="scale", random_state=42)
svm_rbf.fit(X_train, y_train)
```

```
▾ SVC
SVC(random_state=42)
```

```
y_pred_rbf = svm_rbf.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred_rbf)
print("Accuracy:", accuracy)
classification_rep = classification_report(y_test, y_pred_rbf)
print("Classification Report:\n", classification_rep)
confusion_mat = confusion_matrix(y_test, y_pred_rbf)
print("Confusion Matrix:\n", confusion_mat)
sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Blues")
```

```
Accuracy: 0.8945132313668366
Classification Report:
               precision    recall  f1-score   support

           0       0.91      0.98      0.94      7303
           1       0.60      0.21      0.31       935

    accuracy                           0.89      8238
   macro avg       0.75      0.59      0.63      8238
weighted avg       0.87      0.89      0.87      8238

Confusion Matrix:
 [[7176  127]
 [ 742  193]]
<Axes: >
```



```
svm_poly = SVC(kernel="poly", degree=5, coef0=1, C=1)
svm_poly.fit(X_train, y_train)
```

```
▾ SVC
SVC(C=1, coef0=1, degree=5, kernel='poly')
```

```
y_pred_poly = svm_poly.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred_poly)
print("Accuracy:", accuracy)
classification_rep = classification_report(y_test, y_pred_poly)
print("Classification Report:\n", classification_rep)
confusion_mat = confusion_matrix(y_test, y_pred_poly)
print("Confusion Matrix:\n", confusion_mat)
sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Blues")
```

```
    Accuracy: 0.8953629521728574
```