```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix


# Load data
data = pd.read_csv("bill_authentication.csv")


# Features and target
X = data.drop("Class", axis=1)
y = data["Class"]


# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)



# Train an SVM model with linear kernel
svc_model = SVC(kernel='linear')
svc_model.fit(X_train, y_train)
```

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

```
# Predict
y_pred = svc_model.predict(X_test)


# Evaluate
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[155   2]
 [  0 118]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       157
           1       0.98      1.00      0.99       118

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Load data
data = pd.read_csv("bill_authentication.csv")
X = data.drop("Class", axis=1)
y = data["Class"]

# Define combinations of test_size and random_state
settings = [
    (0.2, 0),
    (0.3, 1),
    (0.5, 2)
]

# Loop through each setting and train SVM
for test_size, seed in settings:
    print(f"\n=== Test Size: {test_size}, Random State: {seed} ===")

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)

    # Train model
    model = SVC(kernel='linear')
    model.fit(X_train, y_train)

    # Predict and evaluate
    y_pred = model.predict(X_test)
```

```
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
=== Test Size: 0.2, Random State: 0 ===
Confusion Matrix:
 [[155   2]
 [  0 118]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.99      0.99       157
           1       0.98      1.00      0.99       118

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275


=== Test Size: 0.3, Random State: 1 ===
Confusion Matrix:
 [[233   2]
 [  2 175]]
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.99      0.99       235
           1       0.99      0.99      0.99       177

    accuracy                           0.99       412
   macro avg       0.99      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412


=== Test Size: 0.5, Random State: 2 ===
Confusion Matrix:
 [[381   6]
 [  0 299]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.98      0.99       387
           1       0.98      1.00      0.99       299

    accuracy                           0.99       686
   macro avg       0.99      0.99      0.99       686
weighted avg       0.99      0.99      0.99       686
```

Start coding or generate with AI.