1) Import Libraries

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import cross_val_score, KFold
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.decomposition import PCA

# Load the dataset
data = pd.read_csv('/content/HR_comma_sep.csv')
data.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_la:
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	
3	0.72	0.87	5	223	5	0	1	
4	0.37	0.52	2	159	3	0	1	

Next steps: Generate code with data | View recommended plots | New interactive sheet

2) Preprocess the data

```
le_department = LabelEncoder()
le_salary = LabelEncoder()
data['Department'] = le_department.fit_transform(data['Department'])
data['salary'] = le_salary.fit_transform(data['salary'])
data.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_la:
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	
3	0.72	0.87	5	223	5	0	1	
4	0.37	0.52	2	159	3	0	1	

Next steps: Generate code with data | View recommended plots | New interactive sheet

3) Separate features and target variable

```
X = data.drop('left', axis=1)
y = data['left']

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA to reduce the dataset dimensions while retaining 95% of variance
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)
```

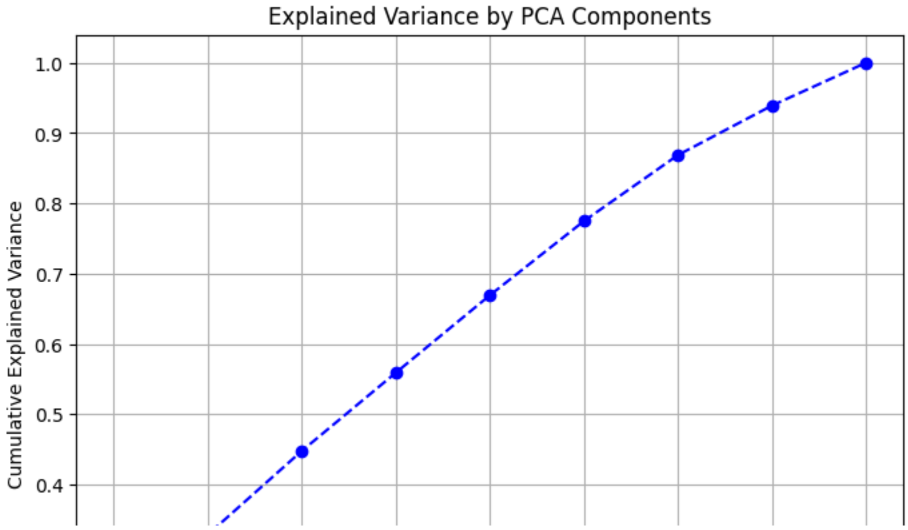
4) Plot explained variance by PCA components

```
explained_variance = np.cumsum(pca.explained_variance_ratio_)
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(explained_variance) + 1), explained_variance, marker='o', linestyle='--', color='b')
plt.title('Explained Variance by PCA Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid()
plt.show()

# Define cross-validation method
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Evaluate Naïve Bayes and SVM using cross-validation on the full dataset
nb_classifier = GaussianNB()
svm_classifier = SVC(kernel='linear', random_state=42)

nb_scores = cross_val_score(nb_classifier, X_scaled, y, cv=kf, scoring='accuracy')
svm_scores = cross_val_score(svm_classifier, X_scaled, y, cv=kf, scoring='accuracy')
```



5) Evaluate Naïve Bayes and SVM using cross-validation on the PCA-reduced dataset

```
nb_scores_pca = cross_val_score(nb_classifier, X_pca, y, cv=kf, scoring='accuracy')
svm_scores_pca = cross_val_score(svm_classifier, X_pca, y, cv=kf, scoring='accuracy')

# Display mean accuracy results
print("Mean Accuracy of Naïve Bayes (Full Dataset):", np.mean(nb_scores))
print("Mean Accuracy of SVM (Full Dataset):", np.mean(svm_scores))
print("\nMean Accuracy of Naïve Bayes (PCA-Reduced Dataset):", np.mean(nb_scores_pca))
print("Mean Accuracy of SVM (PCA-Reduced Dataset):", np.mean(svm_scores_pca))
```

```
Mean Accuracy of Naïve Bayes (Full Dataset): 0.796186351005891
Mean Accuracy of SVM (Full Dataset): 0.7641172168500612

Mean Accuracy of Naïve Bayes (PCA-Reduced Dataset): 0.8674575302878736
Mean Accuracy of SVM (PCA-Reduced Dataset): 0.7641172168500612
```