In [17]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [18]:

```
df = pd.read_csv("WineQt.csv")
df.head()
```

Out[18]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	

In [19]:

```
df.info()
df.isnull().sum()
df.describe()
```

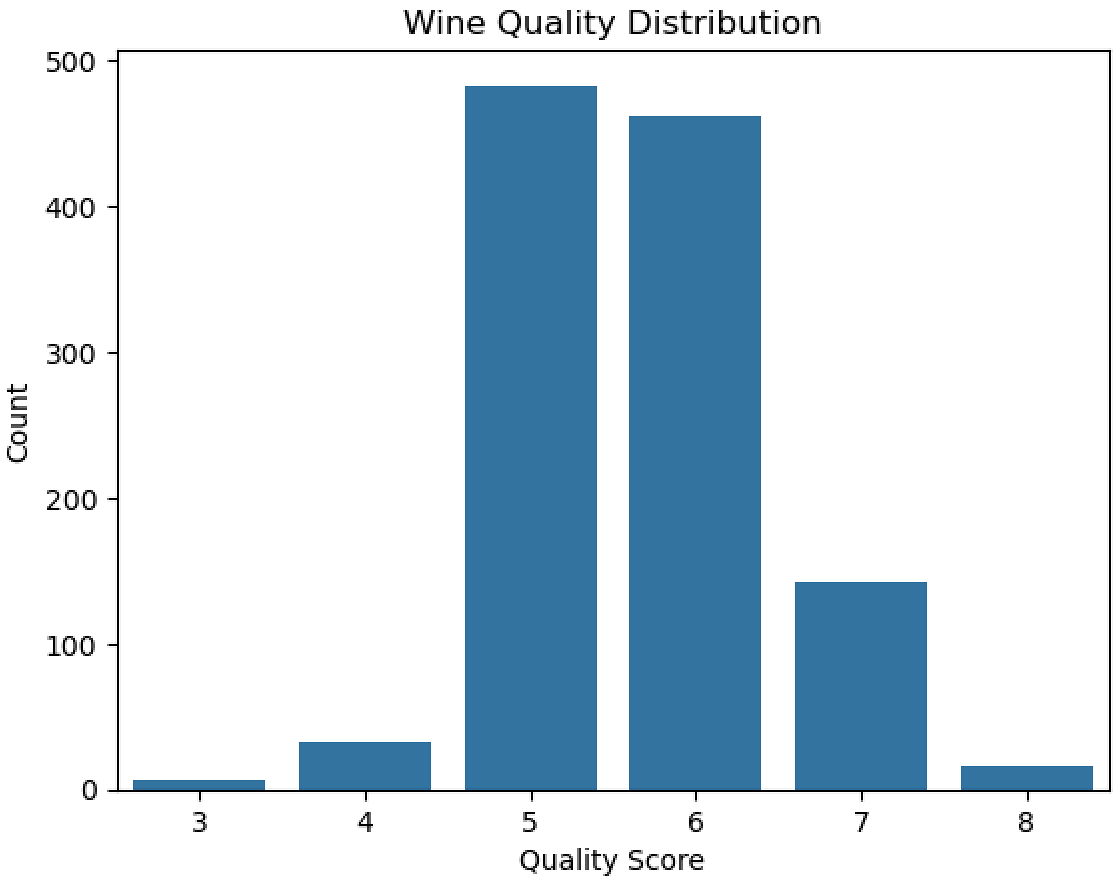
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1143 non-null   float64
 1   volatile acidity      1143 non-null   float64
 2   citric acid           1143 non-null   float64
 3   residual sugar        1143 non-null   float64
 4   chlorides             1143 non-null   float64
 5   free sulfur dioxide   1143 non-null   float64
 6   total sulfur dioxide  1143 non-null   float64
 7   density               1143 non-null   float64
 8   pH                    1143 non-null   float64
 9   sulphates             1143 non-null   float64
 10  alcohol               1143 non-null   float64
 11  quality               1143 non-null   int64  
 12  Id                    1143 non-null   int64  
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
```

Out[19]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	tota
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000	1143
mean	8.311111	0.531339	0.268364	2.532152	0.086933	15.615486	45
std	1.747595	0.179633	0.196686	1.355917	0.047267	10.250486	32
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6
25%	7.100000	0.392500	0.090000	1.900000	0.070000	7.000000	21
50%	7.900000	0.520000	0.250000	2.200000	0.079000	13.000000	37
75%	9.100000	0.640000	0.420000	2.600000	0.090000	21.000000	61
max	15.900000	1.580000	1.000000	15.500000	0.611000	68.000000	289

In [20]:

```
sns.countplot(x='quality', data=df)
plt.title("Wine Quality Distribution")
plt.xlabel("Quality Score")
plt.ylabel("Count")
plt.show()
```



In [21]:

```
na_counts = df.isnull().sum()
print("Missing values per column:\n", na_counts)
```

```
non_numeric_cols = df.select_dtypes(exclude=[np.number]).columns.tolist()
print("Non-numeric columns:", non_numeric_cols)

for col in non_numeric_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')

if df.isnull().sum().sum() > 0:
    df = df.fillna(df.median(numeric_only=True))

print("Quality distribution:\n", df['quality'].value_counts().sort_index())
```

```
Missing values per column:
 fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
Id                      0
dtype: int64
Non-numeric columns: []
Quality distribution:
 quality
3      6
4     33
5    483
6    462
7    143
8     16
Name: count, dtype: int64
```

In [22]:

```
df['quality_binary'] = df['quality'].apply(lambda x: 1 if x >= 6 else 0)
```

In [23]:

```
# Features and target
X = df.drop('quality', axis=1)
y = df['quality']

# Split into train and test
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, stratify=y  # stratify يحافظ على التوزيع
)
```

In [24]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

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

In [25]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# 1) Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print(classification_report(y_test, rf_pred))

# 2) SGD Classifier
sgd_model = SGDClassifier(random_state=42)
sgd_model.fit(X_train, y_train)
sgd_pred = sgd_model.predict(X_test)
print("SGD Accuracy:", accuracy_score(y_test, sgd_pred))
print(classification_report(y_test, sgd_pred))

# 3) Support Vector Classifier
svc_model = SVC(random_state=42)
svc_model.fit(X_train, y_train)
svc_pred = svc_model.predict(X_test)
print("SVC Accuracy:", accuracy_score(y_test, svc_pred))
print(classification_report(y_test, svc_pred))
```

```
Random Forest Accuracy: 0.8646288209606987
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.92      1.00      0.96        97
           6       0.83      0.96      0.89        92
           7       0.72      0.45      0.55        29
           8       0.00      0.00      0.00         3

    accuracy                           0.86       229
   macro avg       0.41      0.40      0.40       229
weighted avg       0.82      0.86      0.83       229

SGD Accuracy: 0.7729257641921398
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.92      0.99      0.96        97
           6       0.84      0.72      0.77        92
           7       0.35      0.52      0.42        29
           8       0.00      0.00      0.00         3

    accuracy                           0.77       229
   macro avg       0.35      0.37      0.36       229
weighted avg       0.77      0.77      0.77       229

SVC Accuracy: 0.8471615720524017
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.91      1.00      0.95        97
           6       0.80      0.95      0.87        92
           7       0.77      0.34      0.48        29
           8       0.00      0.00      0.00         3

    accuracy                           0.85       229
   macro avg       0.41      0.38      0.38       229
weighted avg       0.80      0.85      0.81       229
```

```
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
D:\Anaconda_Set_Up\Lib\site-packages\sklearn\metrics\_classification.py:1565: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

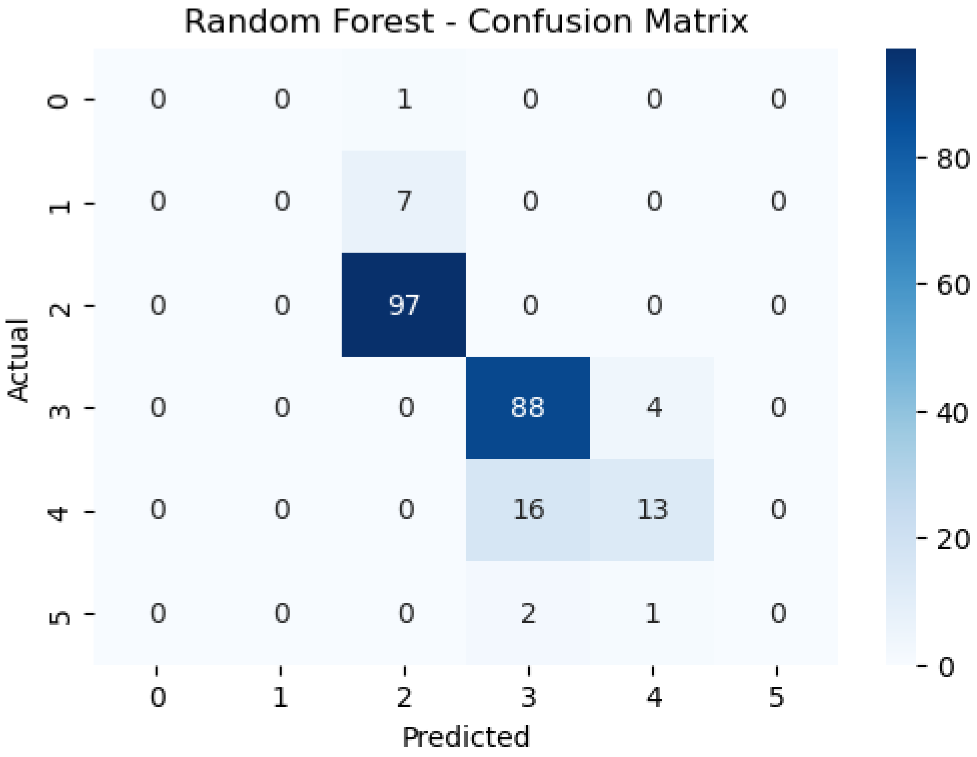
In [26]:

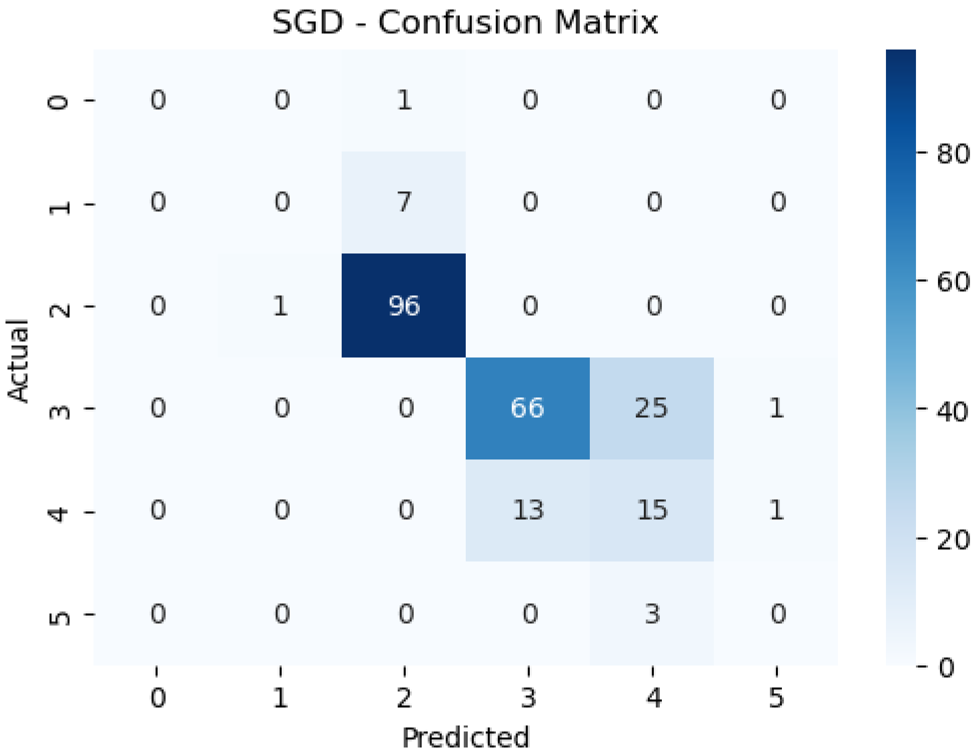
```python
from sklearn.ensemble import GradientBoostingClassifier

gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train, y_train)
gb_pred = gb_model.predict(X_test)

print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_pred))
print(classification_report(y_test, gb_pred))
```

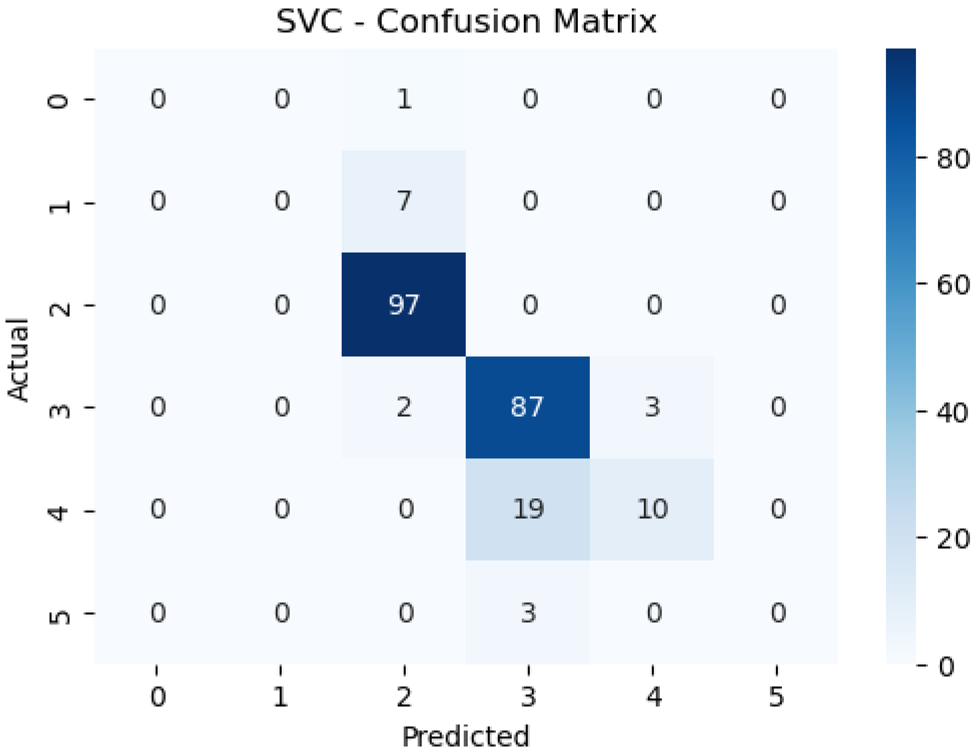
```
Gradient Boosting Accuracy: 0.8646288209606987
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00         7
           5       0.93      0.99      0.96        97
           6       0.85      0.92      0.89        92
           7       0.74      0.59      0.65        29
           8       0.00      0.00      0.00         3

    accuracy                           0.86       229
   macro avg       0.42      0.42      0.42       229
weighted avg       0.83      0.86      0.85       229
```

In [27]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

def plot_confusion(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

plot_confusion(y_test, rf_pred, "Random Forest - Confusion Matrix")
plot_confusion(y_test, sgd_pred, "SGD - Confusion Matrix")
plot_confusion(y_test, svc_pred, "SVC - Confusion Matrix")
```

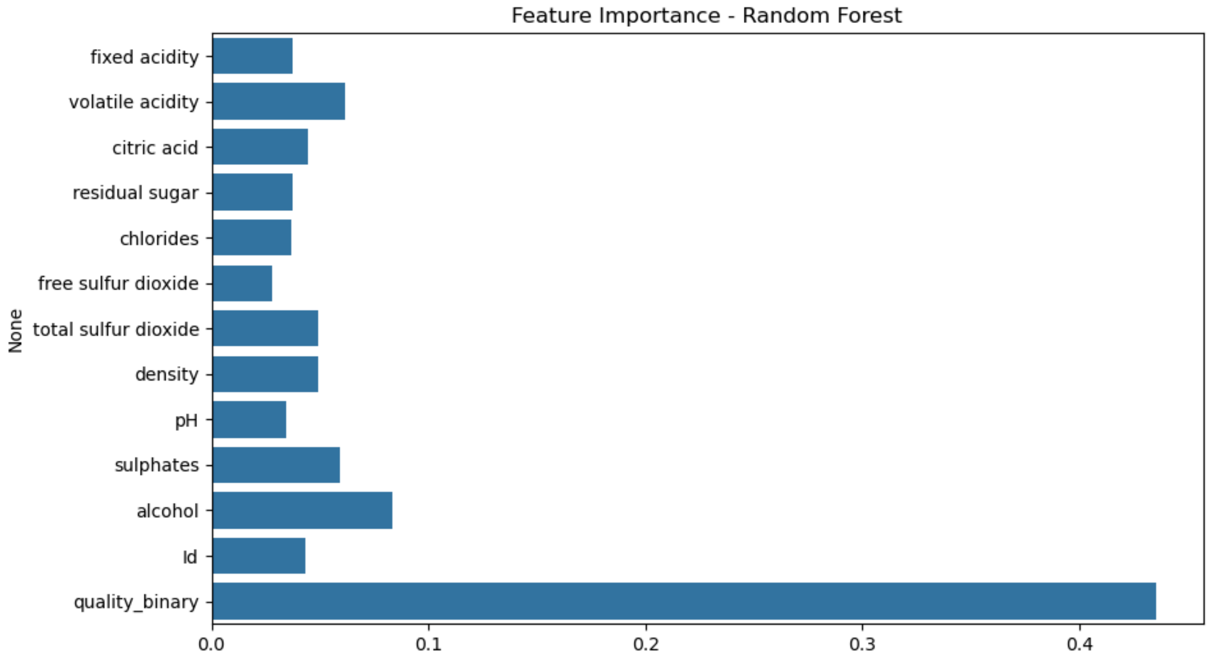






In [28]:

```
importances = rf_model.feature_importances_
features = X.columns

plt.figure(figsize=(10,6))
sns.barplot(x=importances, y=features)
plt.title("Feature Importance - Random Forest")
plt.show()
```



In [29]:

```
results = {
    "RandomForest": accuracy_score(y_test, rf_pred),
    "SGD": accuracy_score(y_test, sgd_pred),
    "SVC": accuracy_score(y_test, svc_pred)
}
print("Model accuracies:", results)
best_model = max(results, key=results.get)
print("Best model:", best_model)
```

```
Model accuracies: {'RandomForest': 0.8646288209606987, 'SGD': 0.7729257641921398, 'S
VC': 0.8471615720524017}
Best model: RandomForest
```