import all the necessary libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from sentence_transformers import SentenceTransformer
import joblib
```

load the data

```
data = pd.read_csv("spam.csv", encoding='latin1')
```

select only the useful columns

```
data = data[['v1', 'v2']]
data.columns = ['label', 'message']
```

converting the labels to numbers (spam/ham) > (0/1)

```
label_encoder = LabelEncoder()
data['label_encoded'] = label_encoder.fit_transform(data['label'])
```

loading the simCSE model which is the smart model to turn text to numbers

```
model = SentenceTransformer('princeton-nlp/sup-simcse-bert-base-uncased')
```

```
WARNING:sentence_transformers.SentenceTransformer:No sentence-transformers model found with name princeton-nlp/sup-simcse-bert-base
```

converting the messages to numerical embeddings

```
print("Creating embeddings. This may take a minute...")
embeddings = model.encode(data['message'].tolist(), show_progress_bar=True)
```

```
Creating embeddings. This may take a minute...
Batches: 100%                                    175/175 [00:10<00:00, 36.99it/s]
```

spliting the data into training and testing

```
X_train, X_test, y_train, y_test = train_test_split(
    embeddings,
    data['label_encoded'],
    test_size=0.2,
    random_state=42,
    stratify=data['label_encoded']
)
```

train the simple model(logistic Regression) > binary classification

```
model_lr = LogisticRegression(max_iter=1000)
model_lr.fit(X_train, y_train)
```

```
▾    LogisticRegression
LogisticRegression(max_iter=1000)
```

testing the model

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

Evaluation results

```
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))
```

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

         ham       0.99      1.00      1.00       966
        spam       0.99      0.97      0.98       149

    accuracy                           0.99      1115
   macro avg       0.99      0.98      0.99      1115
weighted avg       0.99      0.99      0.99      1115
```

Model to save

```
print("Generating sentence embeddings...")
embeddings = model.encode(data['message'].tolist(), show_progress_bar=True)
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train, y_train)
joblib.dump(classifier, 'model.pkl')
print("Model saved as model.pkl")
```

```
Generating sentence embeddings...
Batches: 100%                                   175/175 [00:10<00:00, 37.79it/s]
Model saved as model.pkl
```

Start coding or generate with AI.