```python
import pandas as pd
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\91837\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
```

```python
import pandas as pd

data = {
    'inquiry': [
        "I can't log into my account",
        "How do I reset my password?",
        "My order hasn't arrived yet",
        "I want to return a product",
        "The app crashes when I open it",
        "How to update my billing information?",
        "I received a damaged item",
        "Can I change my delivery address?",
        "The website is very slow",
        "I need help with my subscription"
    ],
    'category': [
        "Account Issues",
        "Account Issues",
        "Order Issues",
        "Order Issues",
        "Technical Issues",
        "Billing",
        "Order Issues",
        "Order Issues",
        "Technical Issues",
        "Subscription"
    ]
}

df = pd.DataFrame(data)
df.to_csv('ticket.csv', index=False)
print("ticket.csv file created successfully.")
```

```
ticket.csv file created successfully.
```

```
df = pd.read_csv('ticket.csv')

print("Sample data:")
print(df.head())
```

```
Sample data:
                              inquiry           category
0     I can't log into my account    Account Issues
1     How do I reset my password?    Account Issues
2     My order hasn't arrived yet      Order Issues
3      I want to return a product      Order Issues
4  The app crashes when I open it  Technical Issues
```

```
# Step 1: Preprocess text

def preprocess_text(text):
    text = str(text).lower()
    stop_words = set(stopwords.words('english'))
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)

df['cleaned_inquiry'] = df['inquiry'].apply(preprocess_text)

# Step 1: Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df['cleaned_inquiry'], df['category'], test_size=0.3,
random_state=42
)

# Step 1: Vectorize text using TF-IDF
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Step 2: Train classifier
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train_tfidf, y_train)
```

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

```
# Step 3: Evaluate model
y_pred = clf.predict(X_test_tfidf)
print("\nModel Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
```

```
Model Accuracy: 0.0

Classification Report:
```

```
                  precision    recall  f1-score   support

  Account Issues       0.00      0.00      0.00       1.0
         Billing       0.00      0.00      0.00       1.0
    Order Issues       0.00      0.00      0.00       0.0
Technical Issues       0.00      0.00      0.00       1.0

        accuracy                           0.00       3.0
       macro avg       0.00      0.00      0.00       3.0
    weighted avg       0.00      0.00      0.00       3.0


E:\Python\Lib\site-packages\sklearn\metrics\_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
E:\Python\Lib\site-packages\sklearn\metrics\_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
E:\Python\Lib\site-packages\sklearn\metrics\_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
E:\Python\Lib\site-packages\sklearn\metrics\_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
E:\Python\Lib\site-packages\sklearn\metrics\_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
E:\Python\Lib\site-packages\sklearn\metrics\_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
# Step 3: Feature importance
feature_names = vectorizer.get_feature_names_out()
for i, category in enumerate(clf.classes_):
    top_features = sorted(
        zip(clf.coef_[i], feature_names), key=lambda x: x[0],
reverse=True
    )[:5]
    print(f"\nTop features for category '{category}':")
    for coef, feat in top_features:
        print(f"  {feat}: {coef:.4f}")
```

```

Top features for category 'Account Issues':
  account: 0.3996
  can: 0.3996
  log: 0.3996
  address: -0.0612
  arrived: -0.0612

Top features for category 'Order Issues':
  address: 0.1836
  arrived: 0.1836
  change: 0.1836
  damaged: 0.1836
  delivery: 0.1836

Top features for category 'Subscription':
  help: 0.3996
  need: 0.3996
  subscription: 0.3996
  address: -0.0612
  arrived: -0.0612

Top features for category 'Technical Issues':
  app: 0.3996
  crashes: 0.3996
  open: 0.3996
  address: -0.0612
  arrived: -0.0612
```

```
# Step 4: Automated response function
def automated_response(text):
    text_clean = preprocess_text(text)
    text_vec = vectorizer.transform([text_clean])
    pred_category = clf.predict(text_vec)[0]

    canned_responses = {
        "Account Issues": "Please try resetting your password using
the 'Forgot Password' link.",
        "Order Issues": "We are looking into your order status and
will update you shortly.",
        "Technical Issues": "Please try reinstalling the app or
clearing your browser cache.",
        "Billing": "You can update your billing information in your
account settings.",
        "Subscription": "For subscription help, please visit your
subscription management page."
    }

    response = canned_responses.get(pred_category, "Thank you for
contacting support. We will get back to you soon.")
    return pred_category, response

# Example usage of automated response
test_inquiry = "I forgot my password and can't login"
category, response = automated_response(test_inquiry)
print(f"\nInquiry: {test_inquiry}")
print(f"Predicted Category: {category}")
print(f"Automated Response: {response}")
```

```
Inquiry: I forgot my password and can't login
Predicted Category: Order Issues
Automated Response: We are looking into your order status and will
update you shortly.
```