```
from google.colab import files
uploaded = files.upload()
```

Choose Files labeled_data.csv

- **labeled_data.csv**(application/vnd.ms-excel) - 2546446 bytes, last modified: 11/27/2020 - 100% done

```
Saving labeled_data.csv to labeled_data.csv
```

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegressionCV, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfTransformer
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline

df = pd.read_csv("labeled_data.csv")


x = [len(df['tweet'][i]) for i in range(df['tweet'].shape[0])]
print('average length of row: {:.3f}'.format(sum(x)/len(x)))
ranges = [0,25, 50, 75, 100, 125, 150]
plt.hist(x, bins = ranges, color=['blue'])
plt.xlabel('Length of Tweets')
plt.ylabel('Number of Tweets')
plt.axis([0, 200, 0, 8000])
plt.grid(True)
plt.show()
# Av no of letters in a tweet
```

```
    average length of row: 85.436
```

```
# Preprocessing
df = df.loc[:, ~df.columns.str.contains('count')]
df.drop(df.columns[df.columns.str.contains('Unnamed',case = False)],axis = 1, inplace = True)
df = df.drop(['class'], axis = 1)
categories = list(df.columns.values)
cvec = CountVectorizer(ngram_range=(1,4), binary = True, analyzer = 'word', stop_words = 'eng
new_df = df.drop(['tweet'], axis = 1)
new_categories = list(new_df.columns.values)
new_categories = list(map(lambda x: str(x), new_categories))
```

```
tfidf_transformer = TfidfTransformer()
text_transformed = cvec.fit_transform(df['tweet'])
text_transformed = tfidf_transformer.fit_transform(text_transformed)
```

```
!pip install transformers
```

```
Collecting transformers
  Downloading https://files.pythonhosted.org/packages/3a/83/e74092e7f24a08d751aa59b37a9
     |████████████████████████████████| 1.3MB 11.2MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: dataclasses; python_version < "3.7" in /usr/local/lib/pyt
Collecting sentencepiece==0.1.91
  Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a78cca907a6ff9
     |████████████████████████████████| 1.1MB 33.0MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
Collecting tokenizers==0.9.3
  Downloading https://files.pythonhosted.org/packages/4c/34/b39eb9994bc3c999270b69c9eea4
     |████████████████████████████████| 2.9MB 39.2MB/s
Requirement already satisfied: protobuf in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from tra
Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (from
Collecting sacremoses
  Downloading https://files.pythonhosted.org/packages/7d/34/09d19aff26edcc8eb2a01bed8e98
     |████████████████████████████████| 890kB 49.2MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from sac
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from sa
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.6/dist-package
Building wheels for collected packages: sacremoses
  Building wheel for sacremoses (setup.py) ... done
  Created wheel for sacremoses: filename=sacremoses-0.0.43-cp36-none-any.whl size=893257
  Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab31a50123635e6fb5e1
Successfully built sacremoses
Installing collected packages: sentencepiece, tokenizers, sacremoses, transformers
Successfully installed sacremoses-0.0.43 sentencepiece-0.1.91 tokenizers-0.9.3 transform
```

```
import torch
from transformers import BertTokenizer, BertModel, BertConfig

# OPTIONAL: if you want to have more information on what's happening, activate the logger as
import logging
#logging.basicConfig(level=logging.INFO)

import matplotlib.pyplot as plt
% matplotlib inline

# Load pre-trained model tokenizer (vocabulary)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
```

```
Downloading: 100%                    232k/232k [00:00<00:00, 566kB/s]
```

```
# Bert
text_embeddings = []
model = BertModel.from_pretrained('bert-base-uncased',
                                  output_hidden_states = True, # Whether the model returns al
                                  )

# Put the model in "evaluation" mode, meaning feed-forward operation.
model.eval()
```

```
Downloading: 100%                    433/433 [00:00<00:00, 793B/s]

Downloading: 100%                    440M/440M [00:10<00:00, 41.1MB/s]

BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(30522, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (1): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
```

```
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (2): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (3): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (4): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
```

```
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (5): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (6): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
```

```
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (7): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (8): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (9): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
```

```
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (10): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
        (value): Linear(in_features=768, out_features=768, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (output): BertSelfOutput(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (intermediate): BertIntermediate(
      (dense): Linear(in_features=768, out_features=3072, bias=True)
    )
    (output): BertOutput(
      (dense): Linear(in_features=3072, out_features=768, bias=True)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
  (11): BertLayer(
    (attention): BertAttention(
      (self): BertSelfAttention(
        (query): Linear(in_features=768, out_features=768, bias=True)
        (key): Linear(in_features=768, out_features=768, bias=True)
```

```
# Bert
df_len = len(df)
print("Length: ", df_len)
for sentence in df['tweet']:
  marked_text = "[CLS] " + sentence + " [SEP]"

  # Split the sentence into tokens.
  tokenized_text = tokenizer.tokenize(marked_text)

  # Map the token strings to their vocabulary indeces.
  indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)
  segments_ids = [1] * len(tokenized_text)
```

```
    tokens_tensor = torch.tensor([indexed_tokens])
    segments_tensors = torch.tensor([segments_ids])

    with torch.no_grad():
      outputs = model(tokens_tensor, segments_tensors)
      hidden_states = outputs[2]

    token_embeddings = torch.stack(hidden_states, dim=0)
    token_embeddings = torch.squeeze(token_embeddings, dim=1)
    token_embeddings = token_embeddings.permute(1,0,2)
    token_vecs = hidden_states[-2][0]

    # Calculate the average of all 22 token vectors.
    sentence_embedding = torch.mean(token_vecs, dim=0)
    text_embeddings.append(sentence_embedding.tolist())
```

```
    Length:  24783
```

```
# Bert
x, y = text_transformed, np.zeros(df_len)
y_dict = {}
for category in new_categories:
  y_dict[category] = y

for category in new_categories:
  for i in range(0, df_len):
    y_dict[category][i] = 1 if df[category][i] > 0 else 0
trained_data = {}
for category in new_categories:
  tweetTrainData, tweetTestData, labelTrainData, labelTestData = train_test_split(text_embedd
  trained_data[category] = (tweetTrainData, tweetTestData, labelTrainData, labelTestData)


# df_len = len(df)
# x, y = text_transformed, np.zeros(df_len)
# y_dict = {}
# for category in new_categories:
#   y_dict[category] = y

# for category in new_categories:
#   for i in range(0, df_len):
#     y_dict[category][i] = 1 if df[category][i] > 0 else 0
# trained_data = {}
# for category in new_categories:
#   tweetTrainData, tweetTestData, labelTrainData, labelTestData = train_test_split(x, y_dict
#   trained_data[category] = (tweetTrainData, tweetTestData, labelTrainData, labelTestData)


# Applying logistic regression and one vs rest classifier
lr =  LogisticRegressionCV(verbose=1, penalty='l2',max_iter=500, class_weight='balanced', mul
ovr = OneVsRestClassifier(lr, n_jobs=-1)
for category in new_categories:
```

```
  ovr.fit(trained_data[category][0], trained_data[category][2])
  accuracy = ovr.score(trained_data[category][1], trained_data[category][3])
  print("Prediction for ", category, " labelled tweets: ", accuracy)
```

```
Prediction for  hate_speech  labelled tweets:  0.8311478716965907
Prediction for  offensive_language  labelled tweets:  0.8333669558200525
Prediction for  neither  labelled tweets:  0.8347791002622554
```

```
# Using pipeline for applying logistic regression and one vs rest classifier
LogReg_pipeline = Pipeline([('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_j
for category in new_categories:
    # Training logistic regression model on train data
    LogReg_pipeline.fit(trained_data[category][0], trained_data[category][2])

    # calculating test accuracy
    prediction = LogReg_pipeline.predict(trained_data[category][1])
    print("Prediction for", category," labelled tweets:", accuracy_score(trained_data[categor
```

```
Prediction for hate_speech  labelled tweets: 0.8819850716158967
Prediction for offensive_language  labelled tweets: 0.8771434335283438
Prediction for neither  labelled tweets: 0.8840024208190438
```

```
# Applying logistic regression only
lr =  LogisticRegressionCV(verbose=1, penalty='l2', solver='lbfgs', max_iter=500, multi_class
for category in new_categories:
  lr.fit(trained_data[category][0], trained_data[category][2])
  accuracy = lr.score(trained_data[category][1], trained_data[category][3])
  print("Prediction for ", category, " labelled tweets: ", accuracy)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  5.5min finished
Prediction for  hate_speech  labelled tweets:  0.880371192253379
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  5.6min finished
Prediction for  offensive_language  labelled tweets:  0.8807746620940085
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: Convergenc
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:  5.1min finished
Prediction for  neither  labelled tweets:  0.8876336493847085
```

```
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier()

#Train the model using the training sets
knn.fit(trained_data[category][0], trained_data[category][2])

#Predict the response for test dataset
y_pred = knn.predict(trained_data[category][1])
accuracy = lr.score(trained_data[category][1], trained_data[category][3])
print("Overall accuracy: ", accuracy)
```

```
Overall accuracy:  0.8876336493847085
```

Double-click (or enter) to edit