src package

Submodules

src.cleaned_loader module

class `src.cleaned_loader.CleanedLoader`(*config*, *csvLoader*) [source]

Bases: `object`

Initializes DataTransformation Object

Parameters:
- **config** (*A dictionary of modifiable variables to be referenced*)
- **csvLoader** (*A util built to facilitate reading CSV files*)

`load_X_Sparse`() [source]

Loads the email messages that have been stemmed and tokenized

`load_cleaned_data`() [source]

Load X Sparse and Cleaned y

`load_cleaned_y`() [source]

Loads the Spam/Ham field that has been stored as boolean values

src.model module

class `src.model.Model`(*config*, *X*, *y*) [source]

Bases: `object`

Initializes Model Object

Parameters:
- **config** (*A dictionary of modifiable variables to be referenced*)
- **X** (*The stemmed, tokenized email contents*)
- **y** (*The Spam/Ham field converted to a boolean value (1/0)*)

`evaluate`() [source]

Evaluate the results of the y_pred predictions against the actual y_test data

`predict()` [source]

Test the trained model against the X_test testing data and store the result in y_pred

`save_pickle_file()` [source]

Save the trained model to a pickle file to be used in the streamlit application

`train_logistic_regression()` [source]

Use the training and testing data to train a logistic regression model

`train_naive_bayes()` [source]

Use the training and testing data to train a Naive Bayes model

`train_random_forest()` [source]

Use the training and testing data to train a Random Forest model

`train_test_split()` [source]

Split the X and y values to be used as training and testing data for the model

src.streamlit_controller module

class **`src.streamlit_controller.StreamlitController`** [source]

Bases: `object`

Initialize the StreamlitController object Load in the external parameters

`SetEmailContent`(*emailContent*) [source]

Take the user’s email content and store it to a local variable :type emailContent: :param emailContent: :type emailContent: The text content of the email to be classified

`get_config()` [source]

Use the yamlLoader utility to fetch the external parameters stored in the params.yaml file

`load_model()` [source]

Load in the most efficient model trained earlier that will be used in the email classification

`predict_email()` [source]

Classify the cleaned-up email as either Spam or Ham using the ML model

`take_words_stem`(*text*) [source]

Take the user's text, remove all stop words, and break it down to just the stems of all alpha words included, then store that to a local variable transformed_content :type text: :param text: :type text: The unedited text found in the email

`tokenize_text`() [source]

Take the transformed_content variable and tokenize it so it can be read by an ML algorithm

`transform_email_content`() [source]

Stem the content of the email and tokenize it to be read by the ML algorithm

src.transform module

class `src.transform.DataTransformation`(*df*, *config*) [source]

Bases: `object`

Initializes DataTransformation Object

Parameters:
- **df** (*The dataframe featuring the data to be transformed*)
- **config** (*A dictionary of modifiable variables to be referenced*)

`declare_x_y_fields`() [source]

Declare the Message field as the X value and the Spam_Bool field as our y value

`make_spam_column_boolean`() [source]

Makes a new column in our dataframe, Spam_Bool that remaps Spam or Ham values to 1 or 0

`remove_na`() [source]

Removes any NA values from the dataframe

`rename_spam_column`() [source]

Removes the Spam/Ham column and replaces it with one just named Spam

`save_data`() [source]

Save the word vectorizer trained earlier, the tokenized words, and the Spam/Ham boolean values to different files

`take_words_stem()` [source]

Iterate through the emails/messages and keep only Alpha words (removing any puctuation or numeric values) Then, replace the words with just their stem for easier reading by the algorithm

`tokenize_text()` [source]

Take the stemmed words and tokenize them so that they can be read by a machine learning model

`transform_data_pipeline()` [source]

Perform the entire data transformation pipeline Remove NA Rename Spam/Ham Column Remap the Spam column to a boolean equivalent

Declare our X and y fields that will be transformed for the model training Stem the words in our X field Tokenize the words in our X field

Save the transformed data for the next run

Module contents