Spam Filter Machine Learning Project Documentation

1. Importing Required Libraries

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
import pandas as pd
import numpy as np
import string
import re
```

In this first part, you import all the necessary libraries. These include libraries for machine learning model creation and evaluation (sklearn), data manipulation (pandas, numpy), and text processing (string, re).

2. Text Preprocessing Functions

```
def to_lower(word):
    return word.lower()

def remove_special_characters(word):
    return word.translate(str.maketrans(dict.fromkeys(string.punctuation)))

def remove_stop_words(words):
    return ' '.join([i for i in words.split() if i not in ENGLISH_STOP_WORDS])

def remove_hyperlink(word):
    return re.sub(r"http\S+|www\S+", "", word)

def preprocess_text(text):
    text = to_lower(text)
    text = remove_special_characters(text)
    text = remove_stop_words(text)
    text = remove_hyperlink(text)
    return text
```

These functions perform preprocessing steps on the text data. The steps include converting text to lowercase, removing special characters and hyperlinks, and filtering out English stop words.

3. Loading and Preprocessing the Data

```
df = pd.read_csv('emails.csv')
df.replace([np.inf, -np.inf], np.nan)
df = df.dropna()
df['text'] = df['text'].apply(preprocess_text)
```

Here, you load the data from a CSV file, handle missing or infinite values, and apply the preprocessing functions to the 'text' column of the DataFrame.

4. Splitting the Data

```
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['spam'],
test_size=0.2, random_state=42)
```

The dataset is split into training and testing sets. The 'text' column is used as the input feature, and 'spam' as the output label.

5. Transforming Text Data

```
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

The text data is transformed into numerical form using TF-IDF vectorization.

6. Training the Classifier

```
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```

A Naive Bayes classifier is trained on the training data.

7. Evaluating the Model

```
predictions = classifier.predict(X_test)
```

The trained model is evaluated on the test data.

8. Classifying a New Email

```
new_email = input("\nEnter the content of the email you want to classify: ")
new_email = preprocess_text(new_email)
new_email_vector = vectorizer.transform([new_email])
spam_probability = classifier.predict_proba(new_email_vector)[0][1]

if spam_probability > threshold:
    print("Likely SPAM.\n")
else:
    print("Likely NOT SPAM.\n")
```

Lastly, the model is used to classify a new email input by the user. The spam probability is calculated, and based on a defined threshold, the email is classified as spam or not spam.