optimizing spam filtering

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[ ] from sklearn.tree import DecisionTreeClassifier
    model = DecisionTreeClassifier()
    model.fit(X_train_res,Y_train_res)
```

```
[ ] from sklearn.ensemble import RandamForestClassifier
    model1 =RandomForestClassifier()
    model1.fit(X_train_res,y_train_res)
```

```
[ ] from sklearn.naive_bayes import MultinomialNB
    model = MultinomialNB()
```

```
[ ] #Fitting the model to the training sets
    model.fit(X_train_res,y_train_res)
```

```
[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
```

```
[ ] #Fitting the model to the training sets
    model = Sequential()
```

```
[ ] X_train.shape
```

optimizing spam filtering

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
X_train.shape
```

```
model.add(Dense(units = X_train_res.shape[1],activation="relu",kernel_initialiser="random_uniform"))
```

```
model.add(Dense(units=100,activation="relu",kernal_initializer="random_uniform"))
```

```
model.add(Dense(units=100,activation="relu",kernal_initializer="random_uniform"))
```

```
model.add(Dense(units=1,activation="sigmoid"))
```

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

```
generator = model.fit(x_train_res,y_train_res,epochs=10,steps_per_epoch=len(x_train_res)//64)
```

```
generator = model.fit(x_train_res,y_train_res,eopchs=10,steps_per_epoch=len(X_train_res)//64)
```

```
y_pred=model.predict(x_test)
y_pred
```

```
y_pr = np.where(y_pred>0.5,1.0)
```

optimizing spam filtering

File Edit View Insert Runtime Tools Help All changes saved

```
model.add(Dense(units=1,activation="sigmoid"))
```

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

```
generator = model.fit(x_train_res,y_train_res,epochs=10,steps_per_epoch=len(x_train_res)//64)
```

```
generator = model.fit(x_train_res,y_train_res,eopchs=10,steps_per_epoch=len(X_train_res)//64)
```

```
y_pred=model.predict(x_test)
y_pred
```

```
y_pr = np.where(y_pred>0.5,1,0)
```

```
y_test
```

```
from sklearn.matrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test,y_pr)
score = accuracy_source(y_test,y_pr)
print(cm)
print('Accuracy Score Is:- ',score*100)
```

```
def new_review(new_review):
    new_review = new_review
```

+ Code + Text

RAM Disk

```
[ ] from sklearn.matrics import confusion_matrix,accuracy_score
    cm = confusion_matrix(y_test,y_pr)
    score = accuracy_source(y_test,y_pr)
    print(cm)
    print('Accuracy Score Is:- ',score*100)
```

```
def new_review(new_review):
    new_review = new_review
    new_review = re.sub('[^a-zA-Z]','',new_review)
    new_review = new_review.lower()
    new_review = new_review.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    new_review = [ps.stem(word) for word in new_review if not word in set(all_stopwords)]
    new_review = ''.join(new_review)
    new_corpus = [new_review]
    new_X_test = cv.transform(new.corpus).toarray()
    print(new_X_test)
    new_y_pred = loaded_model.predict(new_X_test)
    print(new_y_pred)
    new_X_pred = np.where(new_y_pred>0.5,1,0)
    return new_y_pred
new_review = new_review(str(input("Enter new review...")))
```

```
[ ] from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
    cm = confusion_matrix(y_test,y_pred)
```