Machine Learning Project Python Code

December 15, 2021

1 HAM/SPAM Message Classification

Kulpreet Singh - 011771817 Sahil Shrivastava - 011717650
Date of Submission: 12/15/2021 @ 11:00PM PDT
CPTS - 570 MACHINE LEARNING PROJECT
PYTHON CODE

[275]:
```
import pandas as pd
import re
import nltk

from matplotlib import *
import plotly.graph_objects as go
from sklearn.model_selection import cross_val_score
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from wordcloud import WordCloud

nltk.download('stopwords')
df = pd.read_csv('spam.csv')
```

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

[276]:
```
texts = df.iloc[:,:2]
texts.rename(columns={'v1': 'label', 'v2': 'message'}, inplace=True)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py:4441:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[277]:
```
texts['message_len'] = texts.message.apply(len)
```

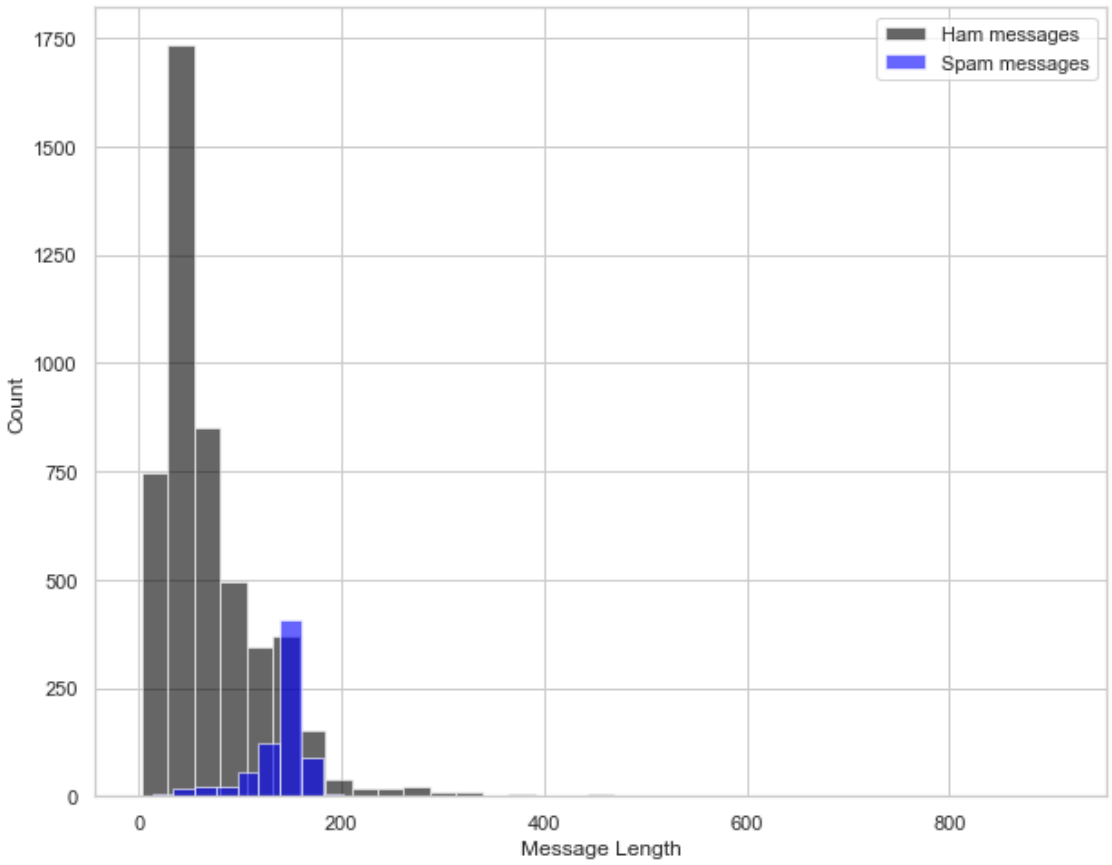
```
<ipython-input-277-545495c63ef1>:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[278]:
```
#This figure shows us the distribution of the ham and spam messages. We can see␣
 ↪that the dataset contains more spam messages.
count = texts.groupby('label')['label'].agg('count').values
fig = go.Figure()
fig.add_trace(go.Bar(
    x=['ham'],
    y=[count[0]],
    name='ham',
    text=[balance_counts[0]],
    textposition='auto',
    marker_color='orange'
))
fig.add_trace(go.Bar(
    x=['spam'],
    y=[count[1]],
    name='spam',
    text=[balance_counts[1]],
    textposition='auto',
    marker_color='pink'
))

fig.show()
```

[279]:
```
#This graph shows us how the higher the lenght of the message the bigger the␣
 ↪chance for it to be a spam message
plt.figure(figsize=(10, 8))
texts[texts.label=='ham'].message_len.plot(bins=35, kind='hist', color='black',
                                           label='Ham messages', alpha=0.6)
texts[texts.label=='spam'].message_len.plot(kind='hist', color='blue',
```

```
                                        label='Spam messages', alpha=0.6)
plt.legend()
plt.xlabel("Message Length")
plt.ylabel("Count")
```

[279]:
```
Text(0, 0.5, 'Count')
```

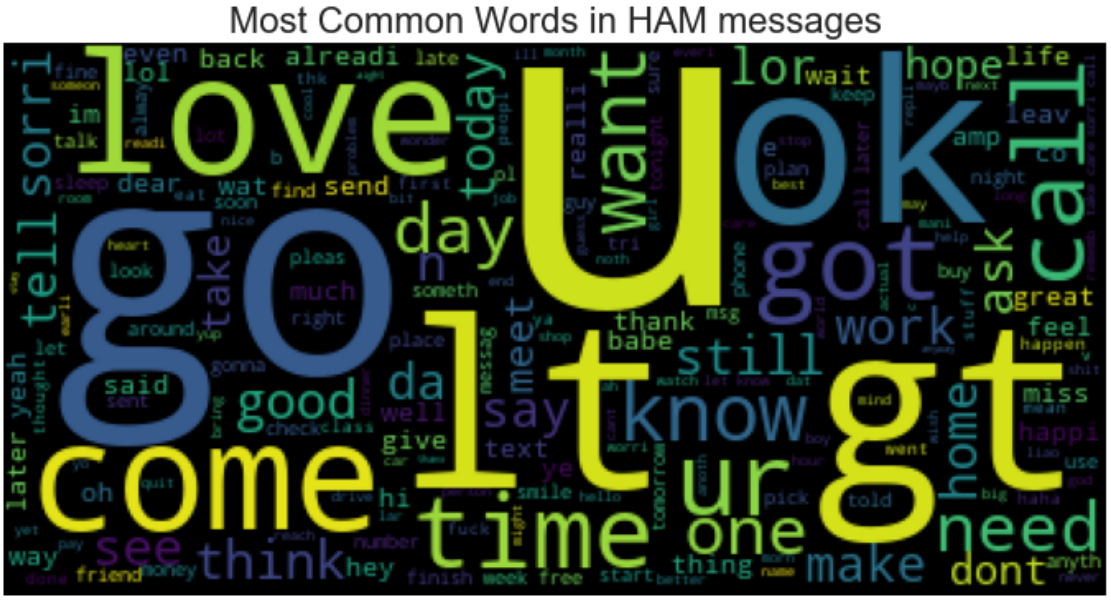


[280]:
```
#This module is going to be used for stemming and will convert words like␣
 ↪revieving, reciever to just reciev. Which will help us classify better
ps = PorterStemmer()
corp = []

#Removing all the useless features from the texts like punctuation words and␣
 ↪stopwords
for i in range(0, len(texts)):
    text = re.sub('[^a-zA-Z]', ' ', texts['message'][i])
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
```

```
    text = text.lower()
    text = text.split()
    text = [ps.stem(word) for word in text if not word in stopwords.
 ↪words('english')]
    text = ' '.join(text)
    corp.append(text)

texts['clean_messages'] = corpus

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2000)
X = cv.fit_transform(corp).toarray()

y=pd.get_dummies(texts['label'])
y=y.iloc[:,1].values


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)
```

```
<ipython-input-280-f38652e510e7>:19: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[281]:
```
wc = WordCloud()
wc.generate(' '.join(text for text in texts.loc[texts['label'] == 'ham',␣
 ↪'clean_messages']))
plt.figure(figsize=(12,8))
plt.title('Most Common Words in HAM messages',
          fontdict={'size': 20})
plt.imshow(wc)
plt.axis("off")
plt.show()
```



[282]:
```
wc = WordCloud()
wc.generate(' '.join(text for text in texts.loc[texts['label'] == 'spam',␣
 ↪'clean_messages']))
plt.figure(figsize=(12,8))
plt.title('Most Common Words in SPAM messages',
          fontdict={'size': 20})
plt.imshow(wc)
plt.axis("off")
plt.show()
```

[283]:
```
#Accuracy to calculate accuracies of individual classifiers.

def acc(y_test,y_pred):
    count = 0
    for i in range(len(y_test)):
        if(y_test[i] == y_pred[i]):
            count+=1
    print(count)
    acc = count/len(y_test)
    return acc
```

[284]:
```
# Training using Naive Bayes Classifier

from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB().fit(X_train, y_train)

prednb =mnb.predict(X_test)
```

[285]:
```
# Training using Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=0).fit(X_train, y_train)

preddtc = dtc.predict(X_test)
```

[286]:
```
# Training using Logistic Regression

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=0).fit(X_train, y_train)

predlr = lr.predict(X_test)
```

[287]:
```
# Training using KNN

from sklearn.neighbors import KNeighborsClassifier
spamKNN = KNeighborsClassifier(n_neighbors = 3).fit(X_train, y_train)

predKNN = spamKNN.predict(X_test)
```

[288]:
```
# Training using Support Vector Machine

from sklearn import svm
spamSVM = svm.SVC(probability = True).fit(X_train,y_train)
predSVM = spamSVM.predict(X_test)
```

[289]:
```
# Training using Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
```

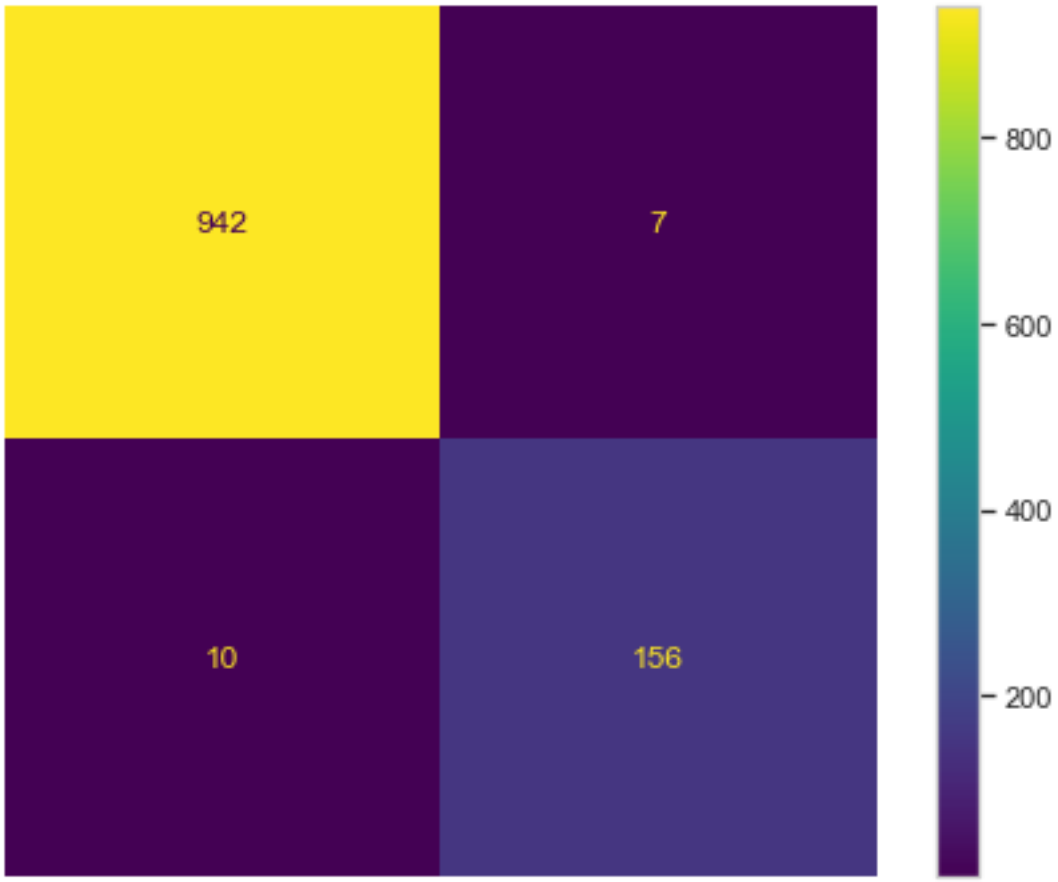
```
spamRF = RandomForestClassifier(random_state = 0).fit(X_train, y_train)
predRF = spamRF.predict(X_test)
```

[290]:
```
# Training using Stochastic Gradient Descent

from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier().fit(X_train,y_train)
predsgd = sgd.predict(X_test)
```

[291]:
```
#Confusion Matrix for Naive Bayes
plt.rcParams["figure.figsize"] = (8, 6)

print(metrics.accuracy_score(y_test, prednb))

cnf_matrix = confusion_matrix(y_test, prednb)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.9847533632286996
```

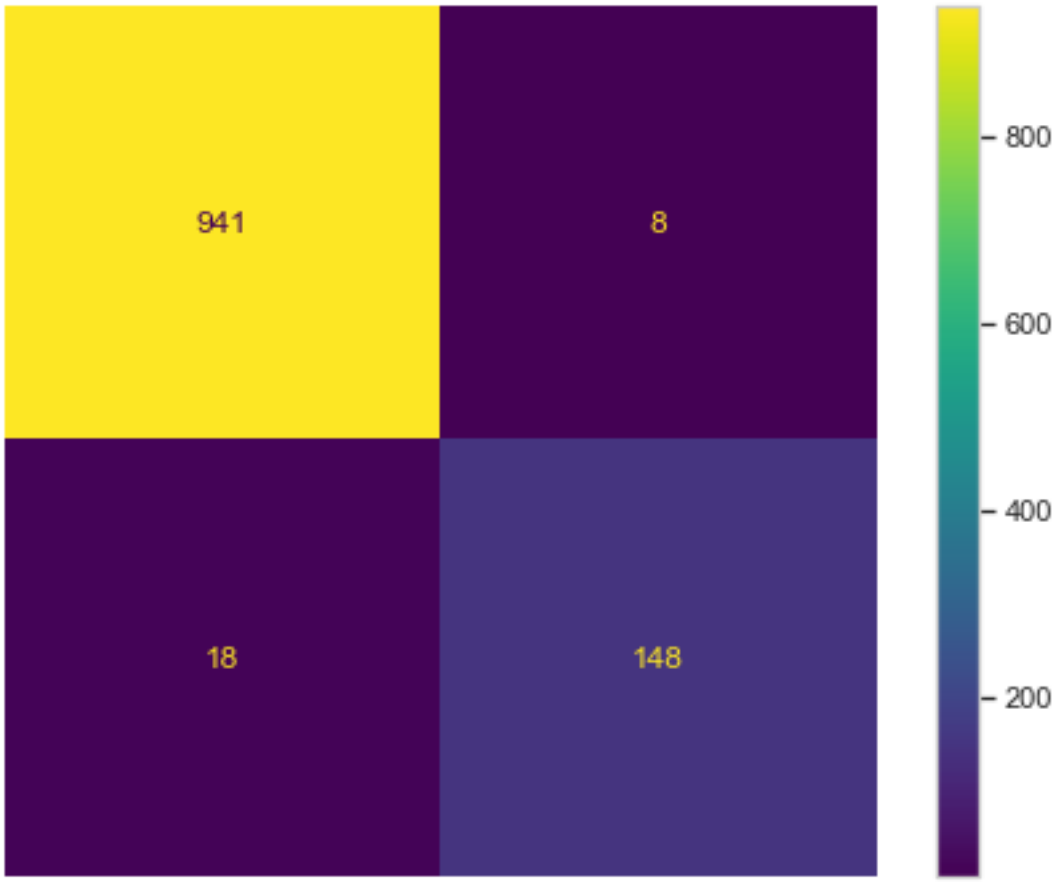
[291]:
```
(-0.5, 1.5, 1.5, -0.5)
```



[292]:
```
#Confusion Matrix for Decision Tree

print(metrics.accuracy_score(y_test, preddtc))

cnf_matrix = confusion_matrix(y_test, preddtc)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.9766816143497757
```

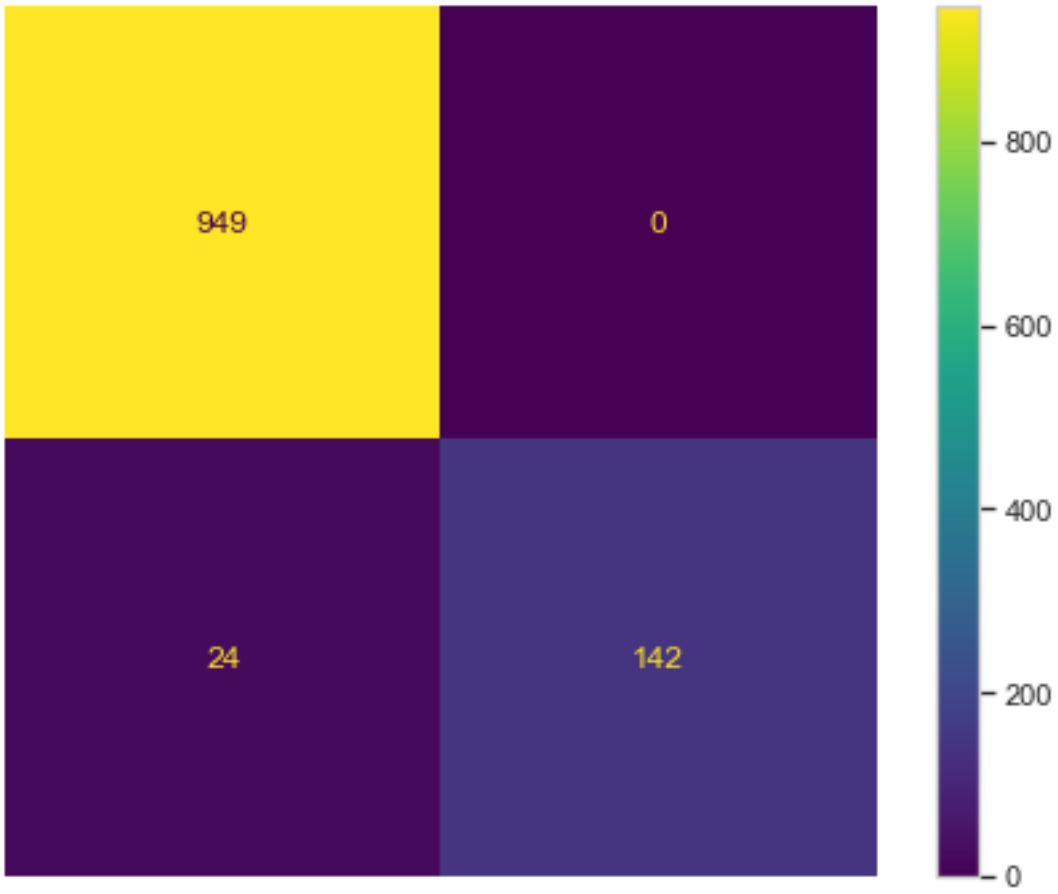
[292]: `(-0.5, 1.5, 1.5, -0.5)`



[293]:
```
#Confusion Matrix for Logistic Regression

print(metrics.accuracy_score(y_test, predlr))

cnf_matrix = confusion_matrix(y_test, predlr)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.97847533632287
```

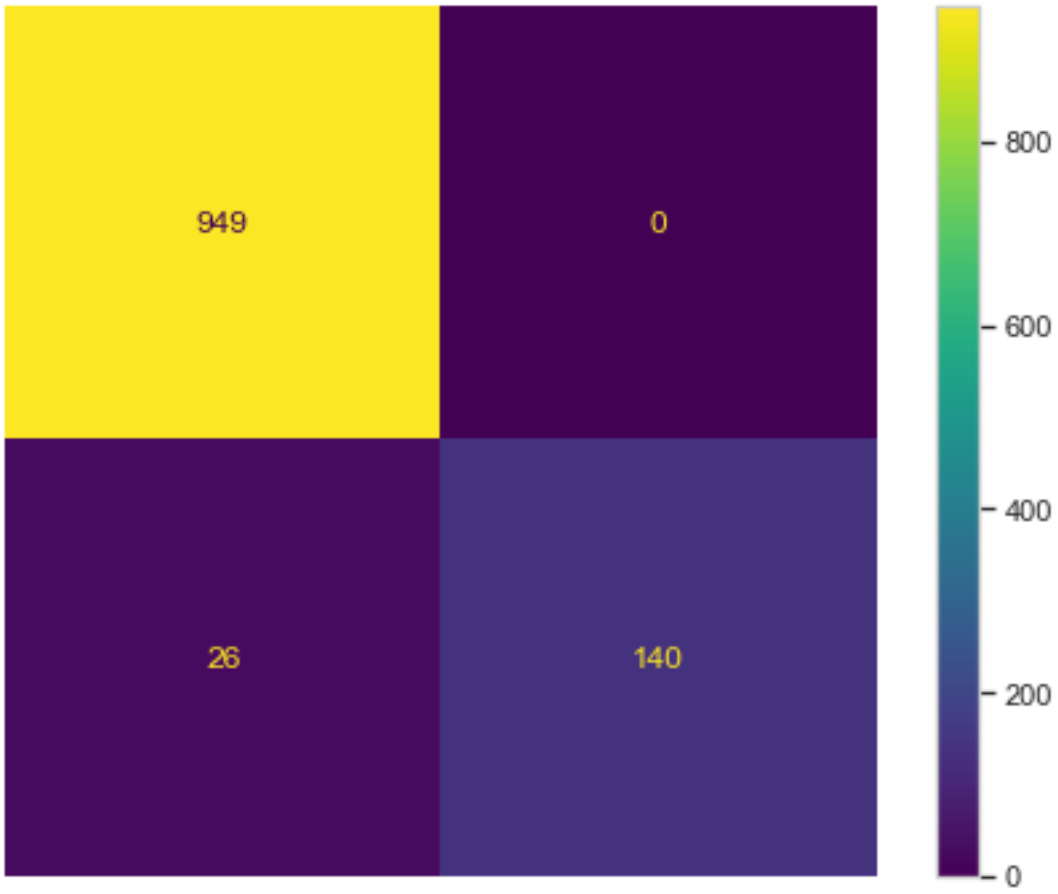
[293]:
```
(-0.5, 1.5, 1.5, -0.5)
```



[294]:
```
#Confusion Matrix for SVM

print(metrics.accuracy_score(y_test, predSVM))

cnf_matrix = confusion_matrix(y_test, predSVM)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.9766816143497757
```

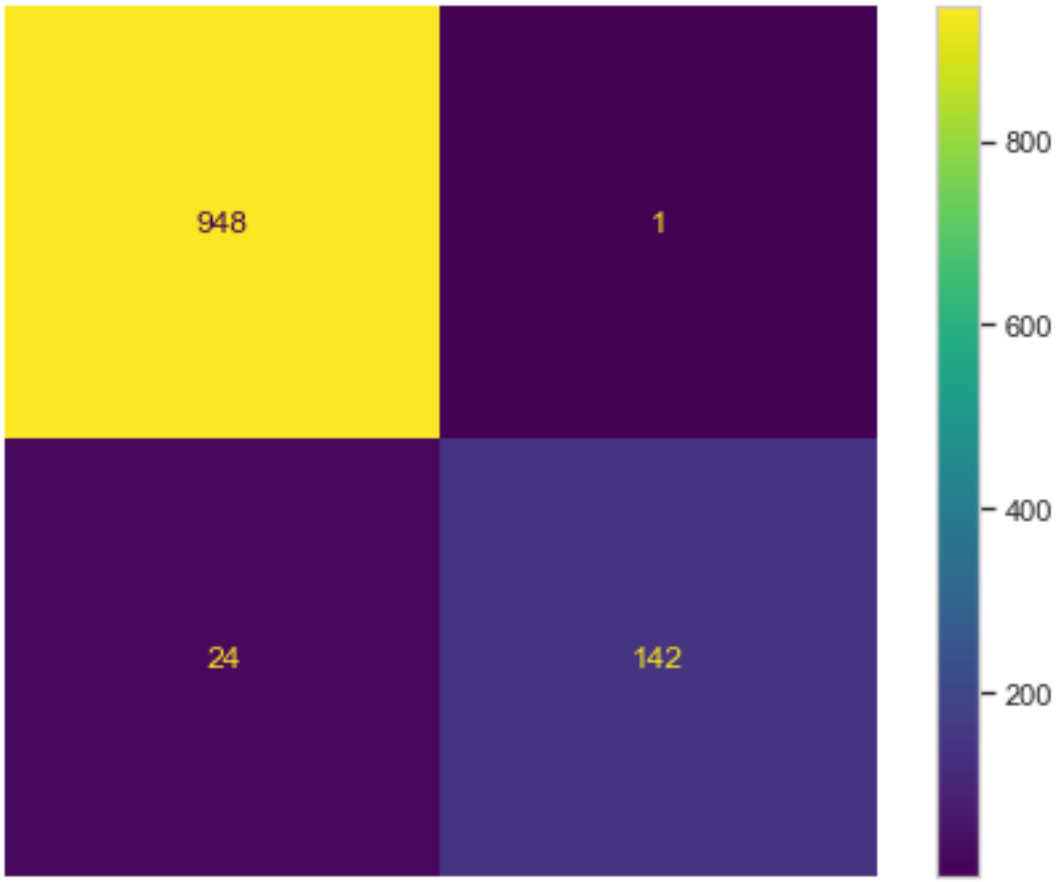
[294]:
```
(-0.5, 1.5, 1.5, -0.5)
```



[295]:
```
#Confusion Matrix for Random Forest

print(metrics.accuracy_score(y_test, predRF))

cnf_matrix = confusion_matrix(y_test, predRF)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.9775784753363229
```

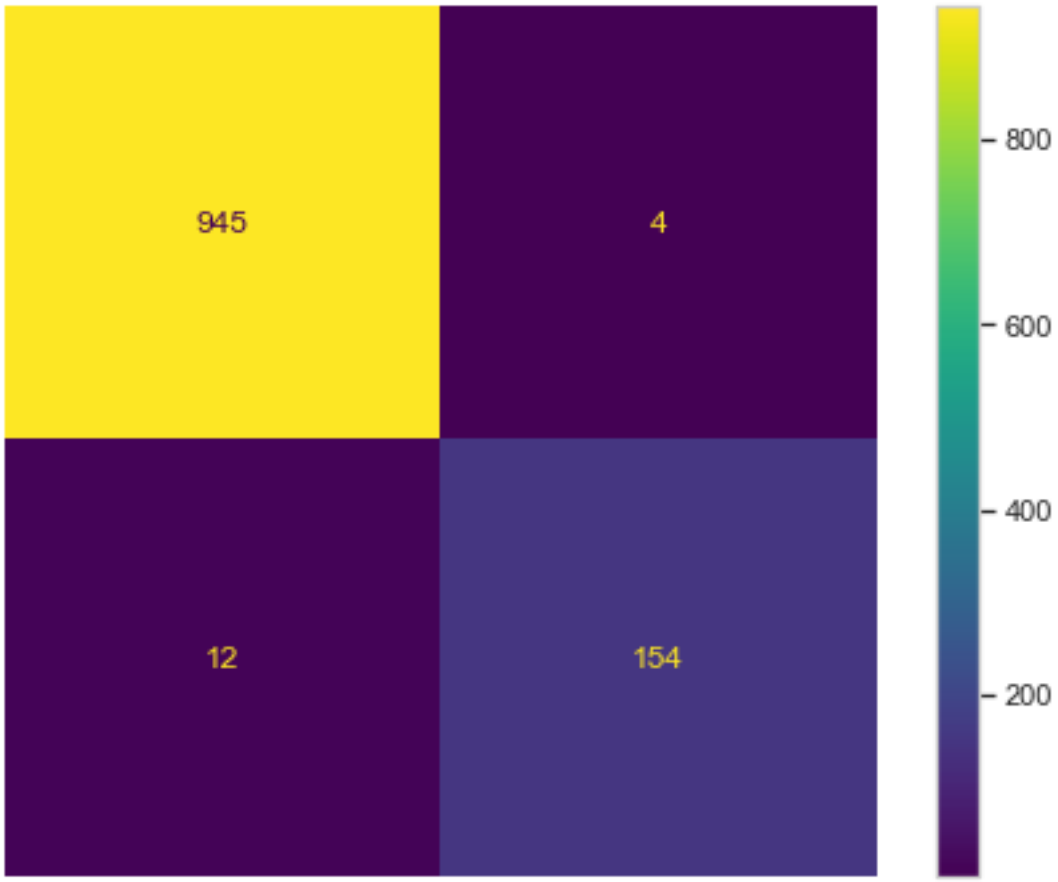
[295]:
```
(-0.5, 1.5, 1.5, -0.5)
```



[296]:
```
#Confusion Matrix for SGD Classifier

print(metrics.accuracy_score(y_test, predsgd))

cnf_matrix = confusion_matrix(y_test, predsgd)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.9856502242152466
```

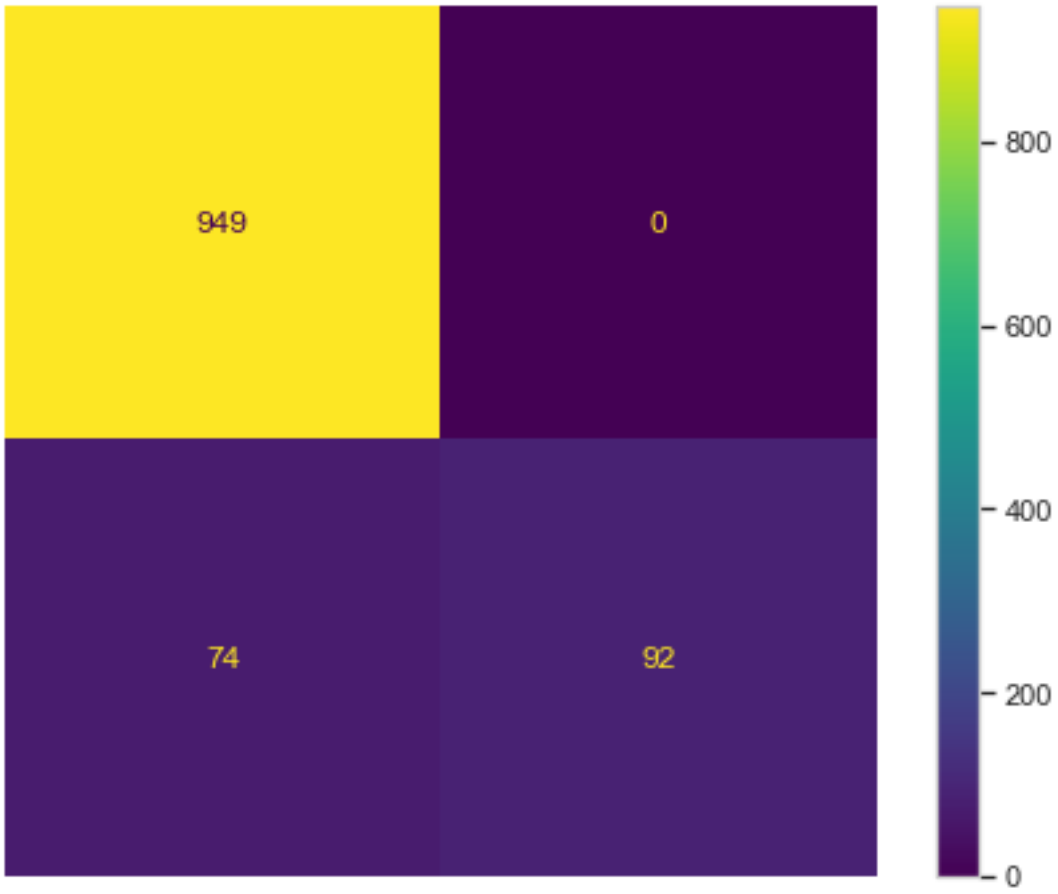
[296]:
```
(-0.5, 1.5, 1.5, -0.5)
```



[297]:
```
#Confusion Matrix for K nearest neighbors

plt.rcParams["figure.figsize"] = (8, 6)

print(metrics.accuracy_score(y_test, predKNN))

cnf_matrix = confusion_matrix(y_test, predKNN)

disp = ConfusionMatrixDisplay(confusion_matrix=cnf_matrix)

disp = disp.plot()

plt.axis('off')
```

```
0.9336322869955157
```

[297]:
```
(-0.5, 1.5, 1.5, -0.5)
```



[298]:
```
texts
```

[298]:
```
     label                                            message  message_len  \
0      ham  Go until jurong point, crazy.. Available only ...          111
1      ham                      Ok lar... Joking wif u oni...           29
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...          155
3      ham  U dun say so early hor... U c already then say...           49
4      ham  Nah I don't think he goes to usf, he lives aro...           61
...    ...                                                ...          ...
5567  spam  This is the 2nd time we have tried 2 contact u...          160
5568   ham              Will _ b going to esplanade fr home?           37
5569   ham  Pity, * was in mood for that. So...any other s...           57
5570   ham  The guy did some bitching but I acted like i'd...          125
5571   ham                        Rofl. Its true to its name           26

                                      clean_messages
0     go jurong point crazi avail bugi n great world...
1                                 ok lar joke wif u oni
2     free entri wkli comp win fa cup final tkt st m...
```

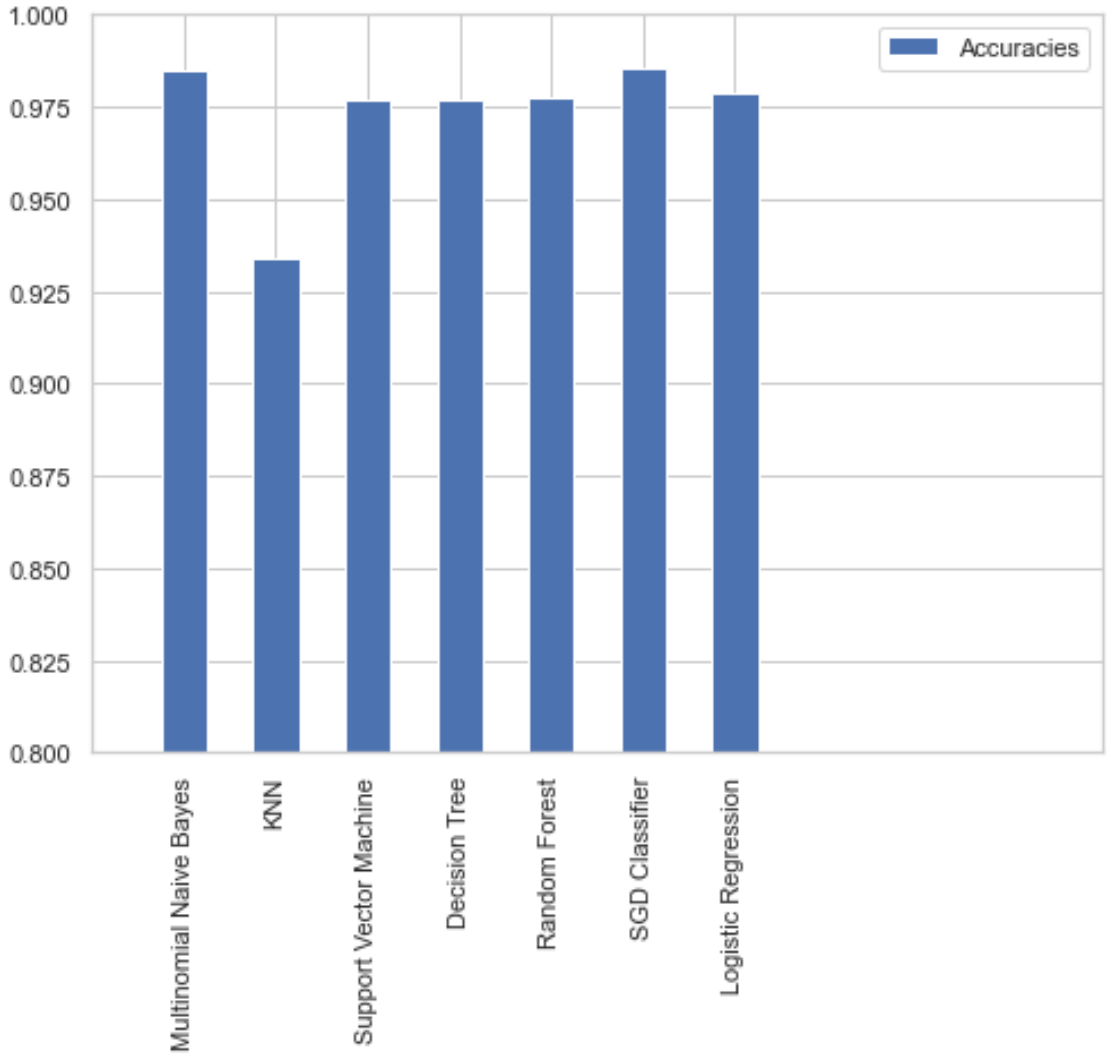
```
3                    u dun say earli hor u c alreadi say
4                   nah think goe usf live around though
...                                                  ...
5567  nd time tri contact u u pound prize claim easi...
5568                              b go esplanad fr home
5569                                 piti mood suggest
5570  guy bitch act like interest buy someth els nex...
5571                                   rofl true name

[5572 rows x 4 columns]
```

[299]:
```
dict =  {'Multinomial Naive Bayes': metrics.accuracy_score(y_test, prednb),␣
↪'KNN': metrics.accuracy_score(y_test, predKNN)
        ,'Support Vector Machine': metrics.accuracy_score(y_test,␣
↪predSVM),'Decision Tree': metrics.accuracy_score(y_test, preddtc)
       ,'Random Forest': metrics.accuracy_score(y_test, predRF),'SGD␣
↪Classifier': metrics.accuracy_score(y_test, predsgd)
       ,'Logistic Regression': metrics.accuracy_score(y_test, predlr)}
```

[300]:
```
df_acc = pd.DataFrame.from_dict(dict,orient='index',columns = ['Accuracies'])
```

[301]:
```
ax = df_acc.plot.bar()
plt.axis([-1, 10, 0.8, 1.0])
```

[301]:
```
(-1.0, 10.0, 0.8, 1.0)
```


1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Accuracies
Multinomial Naive Bayes
KNN
Support Vector Machine
Decision Tree
Random Forest
SGD Classifier
Logistic Regression

[ ]: