SPAM MAIL DETECTION

Project divides into several phases:

1. Data Collection
2. Data Preprocessing
3. Model Training
4. Model Testing
5. Model Evaluation
6. Model Deployment

Importing libraries

```
import warnings
warnings.simplefilter('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Data Collection-

```
data = pd.read_csv("/content/SPAM.csv")
```

```
data.head()
```

	Category	Message	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Next steps: Generate code with data New interactive sheet

Data Pre-Processing-

```
#Checking for Null values
data.isnull().sum()
```

	0
Category	0
Message	0
Unnamed: 2	5522
Unnamed: 3	5560
Unnamed: 4	5566

dtype: int64

```
# Label Encoding
data.loc[data['Category'] == 'spam', 'Category',] = 0
data.loc[data['Category'] == 'ham',  'Category',] = 1
```

```
#Input and Target Value Assignment
X = data['Message']
```

```
Y = data['Category']
```

```
#Data Splitting
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=3)
```

```
#Feature Engineering
from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer(min_df = 1, stop_words='english', lowercase = True)
X_train_features = tf.fit_transform(X_train)
X_test_features  = tf.transform(X_test)
```

Model Training & Testing-

using the following Machine learning models:

1. Logistic Regression
2. Decision Trees
3. K Nearest Neighbors
4. Random Forest

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

```


# Convert Y_train and Y_test to integer type
Y_train_int = Y_train.astype(int)
Y_test_int = Y_test.astype(int)
# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train_features, Y_train_int)
lr_train = lr.predict(X_train_features)
lr_test = lr.predict(X_test_features)
```

```
# Decision Trees

dtrees = DecisionTreeClassifier()
dtrees.fit(X_train_features, Y_train_int)
dt_train = dtrees.predict(X_train_features)
dt_test = dtrees.predict(X_test_features)
```

```
# K Nearest Neighbors

knn = KNeighborsClassifier()
knn.fit(X_train_features, Y_train_int)
knn_train = knn.predict(X_train_features)
knn_test = knn.predict(X_test_features)
```

```
# Random Forest

rf = RandomForestClassifier()
rf.fit(X_train_features, Y_train_int)
rf_train = rf.predict(X_train_features)
rf_test = rf.predict(X_test_features)
```

Model Evaluation-

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

```
# Logistic Regression

lr_train_acc = accuracy_score(Y_train_int, lr_train)
lr_test_acc  = accuracy_score(Y_test_int, lr_test)
lr_precision = precision_score(Y_test_int, lr_test)
lr_recall    = recall_score(Y_test_int, lr_test)
lr_f1        = f1_score(Y_test_int, lr_test)

# Decision Trees

dt_train_acc = accuracy_score(Y_train_int, dt_train)
dt_test_acc  = accuracy_score(Y_test_int, dt_test)
dt_precision = precision_score(Y_test_int, dt_test)
dt_recall    = recall_score(Y_test_int, dt_test)
dt_f1        = f1_score(Y_test_int, dt_test)

# K Nearest Neighbors

knn_train_acc = accuracy_score(Y_train_int, knn_train)
knn_test_acc = accuracy_score(Y_test_int, knn_test)
knn_precision = precision_score(Y_test_int, knn_test)
knn_recall = recall_score(Y_test_int, knn_test)
knn_f1 = f1_score(Y_test_int, knn_test)

# Random Forest

rf_train_acc = accuracy_score(Y_train_int, rf_train)
rf_test_acc = accuracy_score(Y_test_int, rf_test)
rf_precision = precision_score(Y_test_int, rf_test)
rf_recall = recall_score(Y_test_int, rf_test)
rf_f1 = f1_score(Y_test_int, rf_test)
```

```
import pandas as pd

results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Decision Trees', 'K Nearest Neighbors', 'Random Forest'],
    'Train Accuracy': [lr_train_acc, dt_train_acc, knn_train_acc, rf_train_acc],
    'Test Accuracy': [lr_test_acc, dt_test_acc, knn_test_acc, rf_test_acc],
    'Precision': [lr_precision, dt_precision, knn_precision, knn_precision],
    'Recall': [lr_recall, dt_recall, knn_recall, rf_recall],
    'F1-Score': [lr_f1, dt_f1, knn_f1, rf_f1]
})

display(results)
```

```
                 Model  Train Accuracy  Test Accuracy  Precision    Recall  F1-Score
0  Logistic Regression        0.966121       0.962332   0.959000  0.998958  0.978571
1       Decision Trees        1.000000       0.964126   0.972279  0.986458  0.979317
2  K Nearest Neighbors        0.919901       0.905830   0.901408  1.000000  0.948148
3        Random Forest        1.000000       0.982063   0.901408  0.998958  0.989680
```

Next steps: Generate code with results New interactive sheet

```
import pickle
# do it for tfidf and model
with open('tfidf_vectorizer.pkl', 'wb') as file:
    pickle.dump(tf, file)

with open('random_forest_model.pkl', 'wb') as file:
    pickle.dump(rf, file)
```

```
from google.colab import files

files.download('tfidf_vectorizer.pkl')
files.download('random_forest_model.pkl')
```