📘 **Project: SMS Classification (Spam vs Ham)**

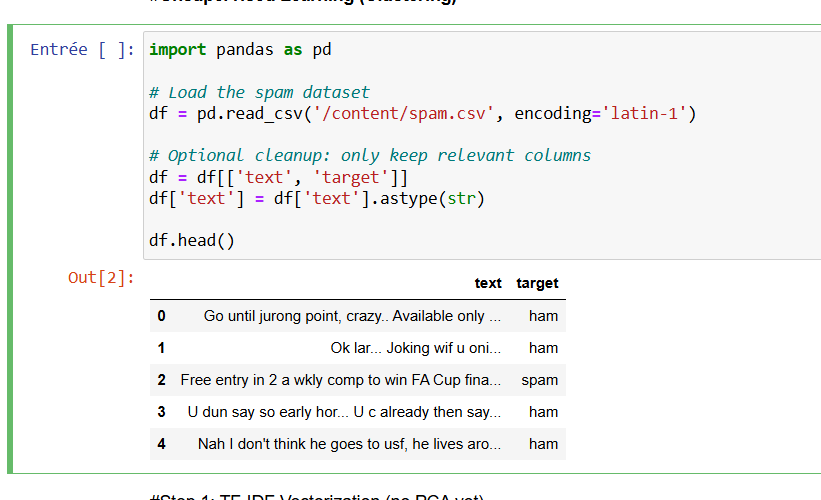
**🎯 Objective**

Automatically classify SMS messages as **"spam"** or **"ham"** (non-spam) using:

* **Unsupervised learning** (clustering)
* **Supervised learning** (classification)

**🗂️ 1. Dataset**

* File used: spam.csv
* Selected columns: text (message), target (ham/spam)



**📌 2. Unsupervised Learning (Clustering)**

**Step 1: TF-IDF Vectorization**

from sklearn.feature\_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop\_words='english', max\_features=1000)

X\_tfidf = vectorizer.fit\_transform(df['text'])

**Step 2: Dimensionality Reduction (PCA)**

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_tfidf.toarray())

**Step 3: K-Means & Hierarchical Clustering**

from sklearn.cluster import KMeans

from scipy.cluster.hierarchy import linkage, fcluster

# K-Means

kmeans = KMeans(n\_clusters=2, random\_state=42)

clusters\_kmeans = kmeans.fit\_predict(X\_tfidf)

# Hierarchical

Z = linkage(X\_tfidf.toarray(), method='ward')

clusters\_hier = fcluster(Z, 2, criterion='maxclust')

**Step 4: Visualization**

import matplotlib.pyplot as plt

# K-Means Clustering

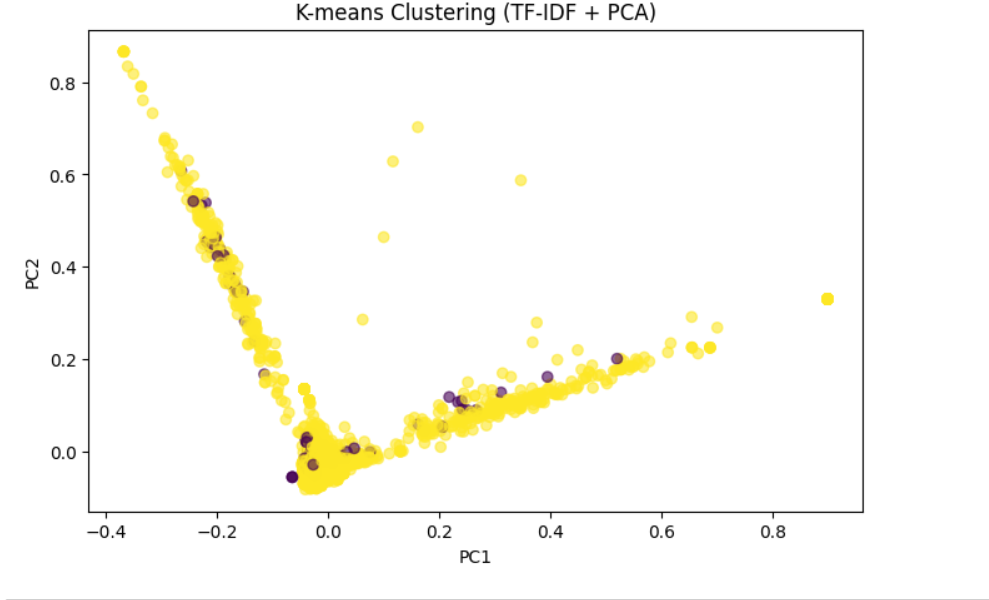
plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=clusters\_kmeans, cmap='viridis', alpha=0.6)

plt.title("K-means Clustering (TF-IDF + PCA)")

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.show()



# Ground Truth Labels

label\_map = {'ham': 0, 'spam': 1}

labels = df['target'].map(label\_map)

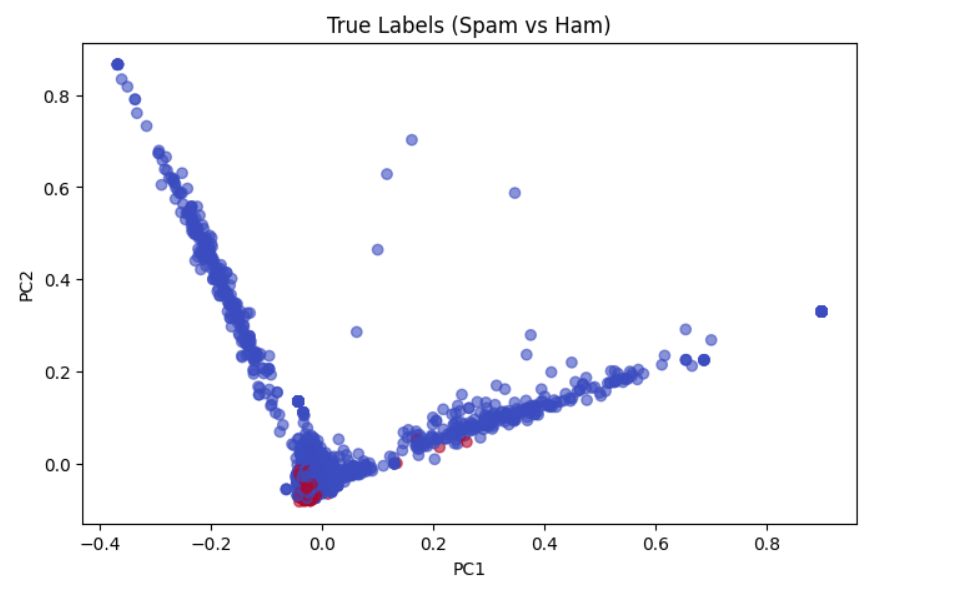
plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=labels, cmap='coolwarm', alpha=0.6)

plt.title("True Labels (Spam vs Ham)")

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.show()



**🧠 3. Supervised Learning (Classification)**

**Step 1: Preprocessing**

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

df['label'] = LabelEncoder().fit\_transform(df['target']) # ham=0, spam=1

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['text'], df['label'], test\_size=0.2, random\_state=42)

**🔤 4. Vectorization with Word2Vec**

**Load Pre-trained Model:**

import gensim.downloader as api

w2v\_model = api.load("word2vec-google-news-300")

**Create Document Vectors:**

def document\_vector(doc):

doc = doc.lower().split()

return np.mean([w2v\_model[w] for w in doc if w in w2v\_model], axis=0)

X\_train\_vec = np.vstack([document\_vector(doc) for doc in X\_train])

X\_test\_vec = np.vstack([document\_vector(doc) for doc in X\_test])

**🔍 5. Classification Models and Results**

**🔸 A. Random Forest**

from sklearn.ensemble import RandomForestClassifier

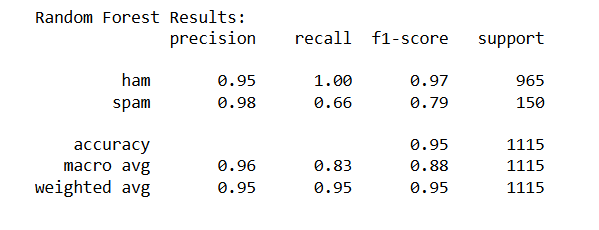
from sklearn.metrics import classification\_report

rf = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf.fit(X\_train\_vec, y\_train)

y\_pred\_rf = rf.predict(X\_test\_vec)

print(classification\_report(y\_test, y\_pred\_rf, target\_names=['ham', 'spam']))



**🔸 B. XGBoost**

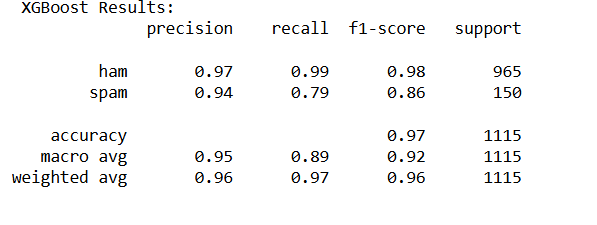
import xgboost as xgb

xgb\_model = xgb.XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss')

xgb\_model.fit(X\_train\_vec, y\_train)

y\_pred\_xgb = xgb\_model.predict(X\_test\_vec)

print(classification\_report(y\_test, y\_pred\_xgb, target\_names=['ham', 'spam']))



**🔸 C. Naive Bayes (TF-IDF Vector Needed)**

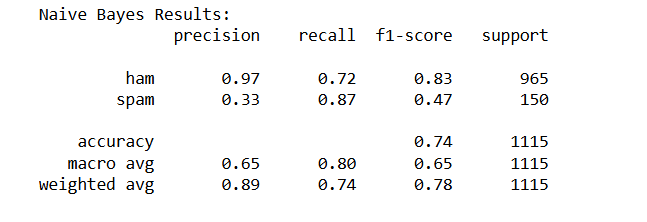
from sklearn.naive\_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X\_train\_vec, y\_train)

y\_pred\_nb = nb.predict(X\_test\_vec)

print(classification\_report(y\_test, y\_pred\_nb, target\_names=['ham', 'spam']))



**📊 6. Confusion Matrix**

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

cm = confusion\_matrix(y\_test, y\_pred\_rf)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['ham', 'spam'])

disp.plot(cmap='Blues')

plt.title("Random Forest Confusion Matrix")

plt.show()

| **Model** | **Accuracy** | **Recall (Spam)** | **F1-score (Spam)** |
| --- | --- | --- | --- |
| **RandomForest** | 0.95 | 0.66 | 0.79 |
| **XGBoost** | 0.97 | 0.86 | 0.86 |
| **Naive Bayes** | 0.65 | 0.87 | 0.47 |

