# Code Report for project.ipynb

## Overview

This notebook implements a machine learning pipeline for spam detection using a variety of classifiers. The project follows a structured workflow of data preprocessing, feature extraction, model training, evaluation, and comparison of multiple machine learning models.

## Libraries & Tools Used

- pandas, numpy for data manipulation  
- matplotlib, seaborn for visualization  
- nltk for text preprocessing  
- TfidfVectorizer for feature extraction  
- sklearn & xgboost for machine learning  
- Various evaluation metrics from sklearn

## Data Preprocessing

Text cleaning involved removing non-alphabetic characters, converting to lowercase, removing stopwords, and applying stemming and lemmatization. This standardized the text for feature extraction.

## Feature Engineering

TF-IDF Vectorizer converted text into numerical features, while Label Encoding transformed target labels.

## Model Training & Evaluation

Several classifiers were used, including Naive Bayes, Random Forest, K-Nearest Neighbors, SVC, Logistic Regression, Gradient Boosting, AdaBoost, Decision Tree, and XGBoost. Models were evaluated using Accuracy, Precision, Recall, F1 Score, Confusion Matrix, and Cross-validation.

## Findings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Precision | Recall | F1 Score | Test Accuracy | Train Accuracy |
| Naive Bayes | 1.000000 | 0.787671 | 0.881226 | 0.977148 | 0.992911 |
| Random Forest | 1.000000 | 0.835616 | 0.910448 | 0.977148 | 0.992911 |
| K-Nearest Neighbors | 1.000000 | 0.390411 | 0.561576 | 0.977148 | 0.992911 |
| SVC | 0.992063 | 0.856164 | 0.919118 | 0.977148 | 0.992911 |
| Linear SVC | 0.984615 | 0.876712 | 0.927536 | 0.977148 | 0.992911 |
| Logistic Regression | 0.982456 | 0.767123 | 0.861538 | 0.977148 | 0.992911 |
| Gradient Boosting | 0.974790 | 0.794521 | 0.875472 | 0.977148 | 0.992911 |
| AdaBoost | 0.864583 | 0.568493 | 0.685950 | 0.977148 | 0.992911 |
| Decision Tree | 0.866197 | 0.842466 | 0.854167 | 0.977148 | 0.992911 |
| XGBoost | 0.961832 | 0.863014 | 0.909747 | 0.977148 | 0.992911 |

## Key Takeaways

- Best Model: LinearSVC (Highest F1 Score and balanced performance)  
- Most Precise: Naive Bayes, Random Forest, and KNN (but recall varies)  
- Worst Recall: K-Nearest Neighbors  
- Most Balanced: XGBoost and Random Forest  
- Weakest Models: AdaBoost and Decision Tree

## Recommendations

- Perform hyperparameter tuning (GridSearchCV, RandomizedSearchCV)  
- Use SHAP or LIME for model explainability  
- Optimize using sklearn Pipelines  
- Apply feature selection or dimensionality reduction  
- Deploy best model in a web app or API