MACHINE LEARNING & NLP

Spam Detection Pipeline

էլեկտրոնային նամակների/հաղորդագրությունների սպամի դասակարգման մոդելի կառուցում Python-ով և Scikit-Learn-ով. կոդի յուրաքանչյուր քայլի մանրամասն վերլուծություն:

1. Գրադարանների ներմուծում

```
import pandas as pd import matplotlib.pyplot as plt import seaborn as sns
```

Pandas (`pd`)

Օգտագործվում է աղյուսակային տվյալների (CSV) բեռնման, մշակման և Dataframe տիպի օբյեկտների հետ աշխատելու համար:

Matplotlib (`plt`)

Ապահովում է գրաֆիկների, դիագրամների և վիզուալ պատկերացումների կառուցման հիմնական գործիքակազմը:

Seaborn (`sns`)

Matplotlib-ի վրա հիմնված գրադարան է, որը հնարավորություն է տալիս ստանալ ավելի գեղեցիկ և վիճակագրական գրաֆիկներ:

2. Տվյալների բեռնում

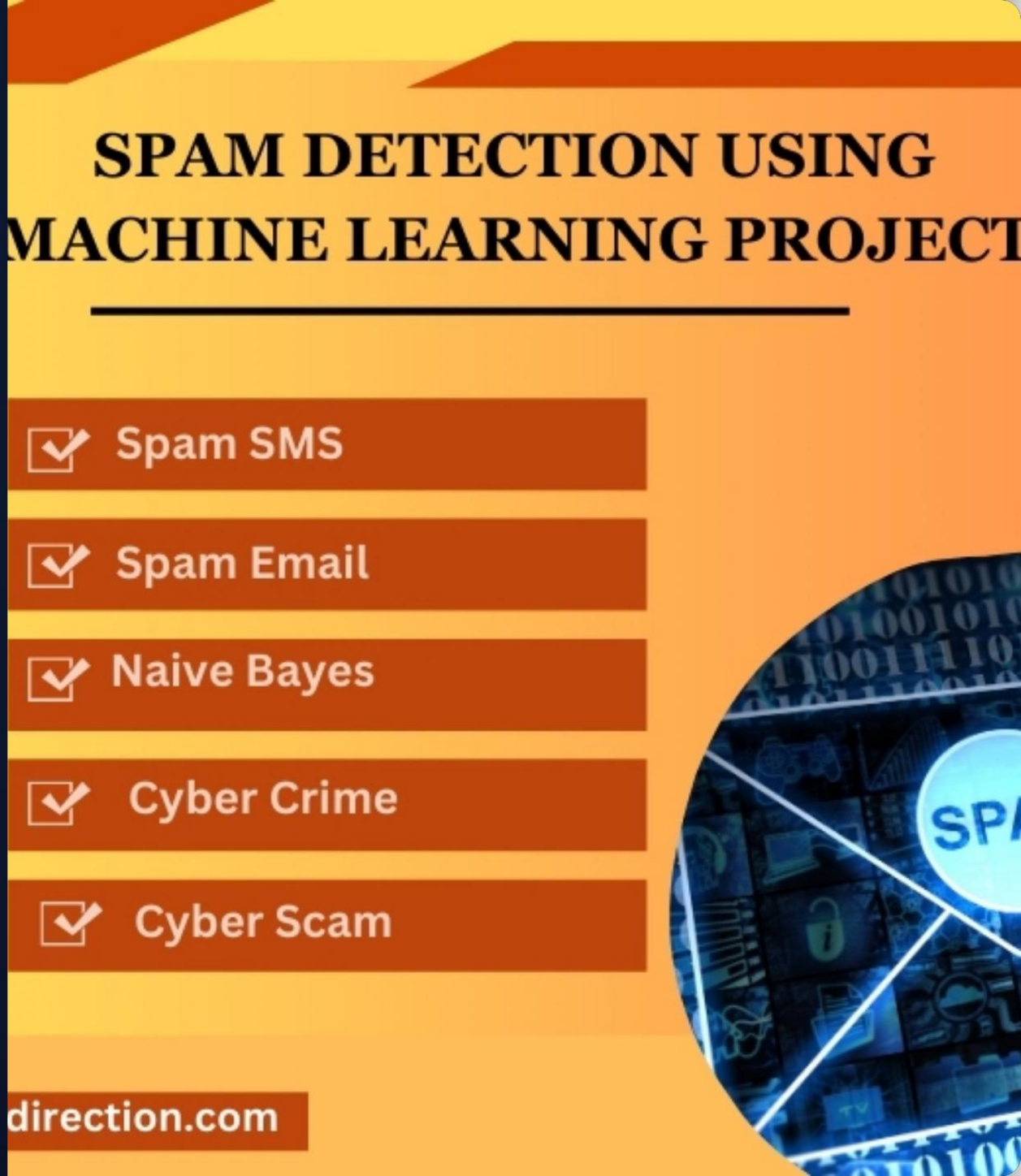
```
df = pd.read_csv("/spam.csv") print(df.head())
print(df.columns)
```

Քայլերի բացատրություն.

`pd.read_csv()` Կարդում է `/spam.csv` ֆայլը և վերածում է Pandas Dataframe-ի:

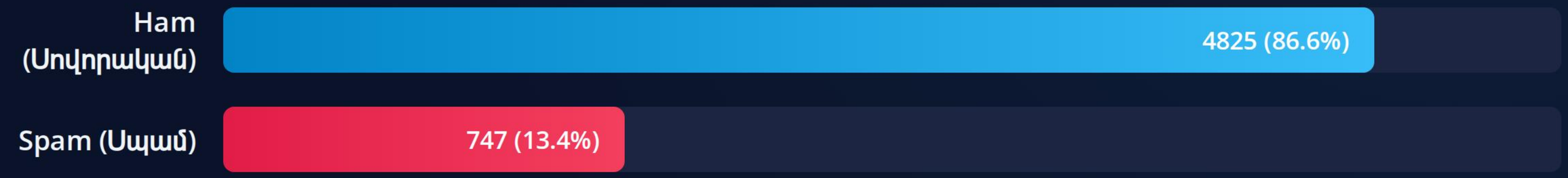
`df.head()` Տպում է տվյալների աղյուսակի առաջին 5 տողերը՝ նախնական ուսումնասիրության համար:

`df.columns` Վերադարձնում է աղյուսակի բոլոր սյուների անվանումները (օր.՝ `label` և `text`):



3. Տվյալների բաշխվածությունը

```
print(df["label"].value_counts()) df["label"].value_counts().plot(kind="bar") plt.title("Spam vs Ham Distribution") plt.xlabel("Label") plt.ylabel("Number of Emails") plt.show()
```



`value_counts()`-ը հաշվում է յուրաքանչյուր դասի քանակը: Գրաֆիկից երևում է, որ տվյալներն ունեն անհավասարակշռված (imbalanced) բաշխվածություն:

4. TF-IDF Վեկտորիզացիա

```
from sklearn.feature_extraction.text import TfidfVectorizer vectorizer = TfidfVectorizer()
```

Ի՞նչ է TF-IDF-ը

TF (Term Frequency): Ցույց է տալիս, թե որքան հաճախ է բառը հանդիպում տվյալ փաստաթղթում/նամակում:

IDF (Inverse Document Frequency): Նվազեցնում է այն ընդհանուր բառերի կշիռը (օր.՝ "the", "is"), որոնք հանդիպում են գրեթե բոլոր նամակներում:

Ինչո՞ւ է դա անհրաժեշտ

Մեքենայական ուսուցման մոդելները չեն կարող ուղղակիորեն աշխատել հում տեքստային տվյալների հետ:

TF-IDF-ը տեքստը վերածում է իմաստալից **թվային մատրիցի**, որտեղ յուրաքանչյուր բառ ստանում է իր կարևորության կշիռը:

5. Հատկանիշների Մատրից

```
X = vectorizer.fit_transform(df["text"])
print(X.shape)
print(vectorizer.get_feature_names_out()[:20])
```

- **`fit_transform()`**: Ստեղծում է բառարանը (vocabulary) և տեքստը վերածում է թվային մատրիցի:
- **`X.shape`**: Ցույց է տալիս մատրիցի չափերը՝ (Նամակների քանակ x Յուրահատուկ բառերի քանակ):
- **`get_feature_names_out()`**: Վերադարձնում է ստացված բառարանի առաջին 20 բառերը:

Text Vectorization Process

6. Տվյալների բաժանում

```
X = vectorizer.fit_transform(df["text"]) y = df["label"] 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=42)
```

80 / 20

Train / Test Հարաբերակցություն

Պարամետրերի բացատրություն.

`test_size=0.2` Տվյալների 20%-ը պահվում է որպես ստուգիչ (Test) բազմություն, իսկ 80%-ն օգտագործվում է մոդելի ուսուցման (Train) համար:

`random_state=42` Ապահովում է տվյալների պատահական բաժանման կրկնելիությունը (reproducibility)՝ ամեն անգամ կոդը աշխատեցնելիս:

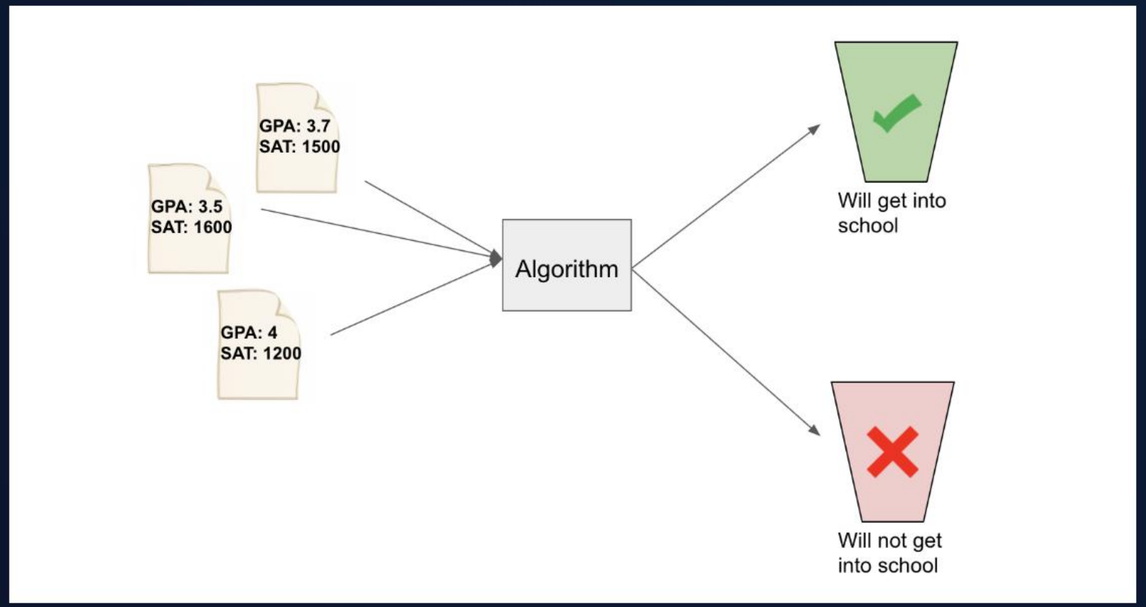
Մոդելի Ուսուցում և Գնահատում

Logistic Regression ալգորիթմի կիրառումը դասակարգման համար

7. Logistic Regression Մոդել

```
from sklearn.linear_model import
LogisticRegression model = LogisticRegression()
model.fit(X_train, y_train)
```

- **`LogisticRegression()`:** Դասակարգման (Classification) հայտնի և արդյունավետ մոդել է, որը հիանալի աշխատում է տեքստային վեկտորների հետ:
- **`model.fit()`:** Մոդելը սովորում է կապ գտնել `X_train` հատկանիշների (բառերի) և `y_train` պիտակների (spam/ham) միջև:



8. Կանխատեսում և Ստուգում

```
y_pred = model.predict(X_test) print(y_pred[:10]) print("Actual: ", y_test.iloc[:10].values)
print("Predicted:", y_pred[:10])
```

N°	Իրական արժեք (y_test)	Կանխատեսված արժեք (y_pred)	Արդյունք
1	ham	ham	✓ Ճիշտ է
2	spam	spam	✓ Ճիշտ է
3	ham	ham	✓ Ճիշտ է
4	ham	ham	✓ Ճիշտ է

`predict(X_test)` *ֆունկցիան օգտագործում է սովորած մոդելը փորձարկման (Test) նոր նամակները դասակարգելու համար:*

9. Ամբողջական Փայփլայն

- **Տվյալների ներբեռնում & EDA:** `read_csv` և `value_counts()` միջոցով տվյալների կառուցվածքի և բաշխվածության ուսումնասիրություն:
- **Տեքստի մշակում (NLP):** `TfidfVectorizer()`-ով տեքստերը վերածվեցին թվային կշիռներ ունեցող հատկանիշների մատրիցի:
- **Տվյալների բաժանում:** `train_test_split()` գործիքով տվյալները բաժանվեցին 80% ուսուցման և 20% ստուգման բազմությունների:
- **Մոդելավորում:** `LogisticRegression()` մոդելի ուսուցում (`fit`) և արդյունքների կանխատեսում (`predict`):

Հարցեր և Քննարկում

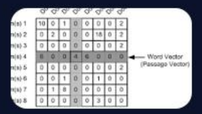
Շնորհակալություն ուշադրության համար

Machine Learning & Natural Language Processing (NLP) Pipeline

Image Sources

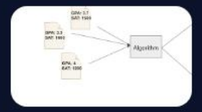
https://www.phddirection.com/wp-content/uploads/2023/11/spam-detection-using-machine-learning-project-3.jpg

Source: www.phddirection.com



https://lh6.googleusercontent.com/GTmNOZ5DkSorxEoATI93xvrBDCCOn0XAGDav8ybPJ0hIkqlk4nimcY9P8SNIeZV1Cf8vnGVAlwawdZ5Fe8kPykKRZbHVUixjSPu1BJdd9DoAdgAVr5VMwhK2oSkXpyDFXunuON-l

Source: www.analyticsvidhya.com



https://towardsdatascience.com/wp-content/uploads/2020/12/1xuDsgnDMiIL8vC39Dao3WQ.png

Source: towardsdatascience.com