Machine Learning

Lab - 3

635 | Vishal Baraiya | 23010101014

📱 Lab: Scikit-Learn Fundamentals (Google Play Store)

Objective: Transition from manual data cleaning to automated Machine Learning preprocessing using Scikit-Learn.

Prerequisites:

- Ensure you have the `googleplaystore_cleaned.csv` file (from the previous lab) in this folder.

1. Load Preprocessed Data

Instruction: Load the dataset you cleaned in the previous lab. This dataset should already have `Installs`, `Price`, and `Reviews` converted to numbers.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("updated.csv")
df.head(5)
```

Out[2]:

	Unnamed: 0	App	Category	Rating	Reviews	Size	Installs	Type
0	0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19000000.0	10000.0	Free
1	1	Coloring book moana	ART_AND_DESIGN	3.9	967	14000000.0	500000.0	Free
2	2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8700000.0	5000000.0	Free
3	3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25000000.0	50000000.0	Free
4	4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2800000.0	100000.0	Free

Intro to Scikit-Learn

What is Scikit-Learn? It is the standard library for Machine Learning in Python. We use it for:

1. **Preprocessing:** Scaling numbers and encoding text.
2. **Modeling:** Training algorithms.
3. **Evaluation:** Checking accuracy.

Task: Import `sklearn` and check the version.

In [3]:
```
import sklearn as sk
```

In [4]:
```
sk.__version__
```

Out[4]:
```
'1.7.2'
```

3. Train_Test_Split

Concept: We split data to prevent "Overfitting". The model learns from the **Train** set and is tested on the **Test** set.

Task:

1. Define `X` (Features: everything except Rating/App) and `y` (Target: Rating).
2. Split the data (80% Train, 20% Test).

In [5]:
```
from  sklearn.model_selection import train_test_split
x = df.drop(['Rating','App'], axis=1)
y = df['Rating']

x_train, x_test, y_train, y_ytest = train_test_split(x ,y , test_size=0.2, random_s
print("Train Size: ",x_train.shape)
print("Test Size: ",x_test.shape)
```

```
Train Size:  (7056, 12)
Test Size:  (1765, 12)
```

4. 📏 Scaling Numerical Data (StandardScaler)

Concept: `Installs` (Millions) are much larger than `Rating` (1-5). We scale them so the model treats them equally.

Task: Use `StandardScaler` on the numerical columns.

In [6]:
```
from sklearn.preprocessing import StandardScaler

num_cols = ['Reviews','Size','Installs','Price']
scalar = StandardScaler()
x_trained_scaled = scalar.fit_transform(x_train[num_cols])
print("scaled data sample")
x_trained_scaled
```

```
scaled data sample
```

Out[6]:
```
array([[-0.04165443, -0.65937412,  0.07693281, -0.06729974],
       [-0.14321967, -0.82894166, -0.15725882, -0.06729974],
       [-0.14237874, -0.49873119, -0.1549285 , -0.06729974],
       ...,
       [-0.14318712,  0.17061436, -0.15724711, -0.06729974],
       [-0.03250245,  1.59855152, -0.04016886, -0.06729974],
       [-0.1007897 ,  0.30448346, -0.1338502 , -0.06729974]],
      shape=(7056, 4))
```

5. 🔠 Encoding Categorical Data

Concept: Models need numbers, not text like "Business" or "Teen".

Method A: Pandas `get_dummies` (Simple)

In [7]:
```
#get_dummies
dummies = pd.get_dummies(x_train['Content Rating'])
```

```
dummies.head()
```

Out[7]:

	Adults only 18+	Everyone	Everyone 10+	Mature 17+	Teen	Unrated
3254	False	True	False	False	False	False
4353	False	True	False	False	False	False
786	False	False	True	False	False	False
6149	False	False	False	False	True	False
449	False	False	False	False	True	False

Method B: Sklearn `OneHotEncoder` (Professional)

In [8]:
```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(handle_unknown='ignore')

cat_encoded = encoder.fit_transform(x_train[['Category']])

print("Encoded Shape : ",cat_encoded.shape)
```

```
Encoded Shape :  (7056, 33)
```

6. 🚀 The Full Pipeline: ColumnTransformer

Concept: Instead of doing steps 4 and 5 manually, we wrap them in one object.

Task: Create a `ColumnTransformer` that Scales numerical data AND Encodes categorical data at the same time.

In [9]:
```
from sklearn.compose import ColumnTransformer
```

In [10]:
```
numeric_features = ['Reviews', 'Size', 'Installs', 'Price']
categorical_features = ['Category', 'Content Rating']
```

In [11]:
```
preprocessor = ColumnTransformer(
    transformers=[
    # ('name',Tranformer(),columns)
    ('num',StandardScaler(),numeric_features),
    ('cat',OneHotEncoder(handle_unknown='ignore'),categorical_features)
    ]
)
```

In [12]:
```
from sklearn.pipeline import Pipeline

pipeline = Pipeline(steps=[
    ('preprocess', preprocessor)
])
```

In [13]:
```
from sklearn import set_config
set_config(display='diagram')
```

```
set_config
```

Out[13]:
```
<function sklearn._config.set_config(assume_finite=None, working_memory=None, prin
t_changed_only=None, display=None, pairwise_dist_chunk_size=None, enable_cython_pa
irwise_dist=None, array_api_dispatch=None, transform_output=None, enable_metadata_
routing=None, skip_parameter_validation=None)>
```

In [14]:
```
df.to_csv('updated2.csv')
```

In [ ]: