# feature-engineering-101

March 24, 2024

## 1 This notebook is made for a live coding session

### 1.1 Imports

[ ]:
```
# Data processing library
import pandas as pd

# Directory and file manipulation library
import os

# Numerical python for our mathematical functions
import numpy as np

# As our primary Machine learning library
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA

# For Data Visualisation
import matplotlib.pyplot as plt

# We'll also import seaborn, a Python graphing library
import warnings # current version of seaborn generates a bunch of warnings that␣
 ↪we'll ignore
warnings.filterwarnings("ignore")
import seaborn as sns
sns.set(style="white", color_codes=True)

# Let's run this (CTRL + Enter)

"Finished importing libraries"
```

### 1.2 Let's define Our Directories and Files

[ ]:
```
BASE_DIR = "/kaggle/input"
DATA_DIR = os.path.join(BASE_DIR, "fertility-data")

data_file = os.path.join(DATA_DIR, "FertilityData.csv")
print(f"Path to the initial file containing all the data: {data_file}")
```

### 1.3 Preprocessing

#### 1.3.1 Let's load the csv into our dataframe variable

[ ]:
```
data = pd.read_csv(data_file, encoding= "Latin1") # Read as a dataframe
"Done :>"
```

#### 1.3.2 How much null data do we have?

[ ]:
```
# Check for missing values in each column and calculate the sum
missing_data = data.isna().sum()

# Display the missing data for each column
print("Missing data for each column:")
print(missing_data)
```

#### 1.3.3 Replacing them with the median

[ ]:
```
# Iterate over each column in the BHCG DataFrame
for column in data.columns:
    # Calculate the median of the column (ignoring NaN values)
    column_median = data[column].median()
    # Replace NaN values with the mean of the column
    data[column].fillna(column_median, inplace=True)
print(f"Finished replacing the null values of each column with the median of␣
 ↪that column")
```

[ ]:
```
missing_data = data.isna().sum()

# Display the missing data for each column
print("Missing data for each column:")
print(missing_data)
```

#### 1.3.4 Our data is now ready to use. Let's take a look at it

[ ]:
```
data.head()
```

[ ]:
```
# Let's use our visualization libraries to visualize the correlation of our data
# Correlation basically shows us the relationship between the data
```

```
# Data without the output column
input_columns = data.columns[:-1]
columns = list(input_columns[0:])

# Create pair plot
sns.pairplot(data[columns], diag_kind='kde')
```

# 2 First Feature Engineering Technique

## 2.1 Normalizing the data using min-max scaling (z-score normalization)

[ ]:
```
# Initialize the MinMaxScaler imported from SKLearn
scaler = MinMaxScaler()

# Normalize the content of the DataFrame between 0 and 1
normalized_data = scaler.fit_transform(data)

# Convert the normalized array back to a DataFrame
data = pd.DataFrame(normalized_data, columns=data.columns)

# Display the normalized DataFrame
data.head()
```

### 2.1.1 Let's show the correlation of this scaled data

[ ]:
```
# Let's use our visualization libraries to visualize the correlation of our data
# Correlation basically shows us the relationship between two columns

# Data without the output column
input_columns = data.columns[:-1]
columns = list(input_columns[0:])

# Create pair plot
# sns.pairplot(data[columns], diag_kind='kde')
```

# 3 Second Feature Engineering Technique

## 3.1 Principle Component Analysis PCA

[ ]:
```
# Perform PCA
pca = PCA(n_components=5)  # Specify the number of components to retain
data_pca = pca.fit_transform(data[columns])

# Optional: Explained Variance Ratio
explained_variance_ratio = pca.explained_variance_ratio_
```

```
print("Explained Variance Ratio:", explained_variance_ratio)
print("Total Explained Variance Ratio:", sum(explained_variance_ratio))

# Construct dataframe with PCA components
pca_columns = [f"PC{i+1}" for i in range(data_pca.shape[1])]
data_pca_df = pd.DataFrame(data=data_pca, columns=pca_columns)

# Concatenate original dataframe with PCA components
data_with_pca = pd.concat([data, data_pca_df], axis=1)

data_with_pca.head()
```

## 4 Thankyou for listening