Data Cleaning Full Course

Data Cleaning Using Pandas

import pandas as pd

Load the Excel File

```
df = pd.read_excel("Customer.xlsx")
df
```

Drop Duplicates

```
df = df.drop_duplicates()
df
```

Drop Columns

```
df = df.drop(columns=["Not_Useful_Column"])
df
```

Strip in Columns

lstrip for striping from the left, rstrip will strip from the right

```
df["Last_Name"] = df["Last_Name"].str.lstrip("...")
df
```

Strip all at once

```
df["Last_Name"] = df["Last_Name"].str.strip("_/.")
df
```

Replace in Columns

```
df["Phone_Number"] = df["Phone_Number"].str.replace('[^a-zA-Z0-9]','', regex=True)
df["Phone_Number"] = df["Phone_Number"].str.replace("nan","")
df
```

What the code will do is, it will replace the specific data with an empty string if 1. it is not a lower case alphabet (a-z) 2. it is not an upper case alphabet (A-Z) 3. it is not a number (0 - 9) If the data is not alphanumeric, replace it with an empty string

Convert it to string

```
df["Phone_Number"] = df["Phone_Number"].astype(str)
```

Make it a standard format

```
df["Phone_Number"] = df["Phone_Number"].apply(lambda x: f"{x[:3]}-{x[3:6]}-{x[6:]}" if len(x
```

Split Based On Columns

```
df[["First","Second","Third"]] = df["Address"].str.split(',', n=2, expand=True)
df
```

Remove all the None and NaN values

```
df = df.fillna('')
df
```

Drop or remove specific Data with Empty Phone Number

```
for x in df.index:
    if df.loc[x, "Phone_Number"] == "":
        df.drop(x, inplace=True)
```

Reset The Clean Data By Index

```
df = df.reset_index(drop=True)
df = df.reset_index(drop=True)
df
```

Save The Final Data

```
df.to_excel("CleanData.xlsx", index=False)
```

Machine Learning Full Course

```
import pandas as pd

# import the Decision tree algorithm
from sklearn.tree import DecisionTreeClassifier as DTC

# Train and Split The Data
from sklearn.model_selection import train_test_split as TTS

# Check the Accuracy Score
from sklearn.metrics import accuracy_score as Scores

# Read the csv
music_data = pd.read_csv("music.csv")
```

```
# Set the Inputs by dropping the column name genre
X = music_data.drop(columns=["genre"])

# Set the Output by selecting only the genre
Y = music_data["genre"]

# Split the dataset into test and train, 20% of the dataset will be used for training
X_Train, X_Test, Y_Train, Y_Test = TTS(X, Y, test_size=0.2)

# Initialise the Function
model = DTC()

# Train the model
model.fit(X_Train, Y_Train)

# Grab the predicted Values, It will try to predict the Y_Test values
predictions = model.predict(X_Test)

# Check the accuracy of the scores
TheScore = Scores(Y_Test, predictions)
TheScore
```

Model Persistence

Train the Model and Save the Trained Model to a File

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier as DTC
import joblib  # Import joblib directly

# Read the CSV file
music_data = pd.read_csv("music.csv")

# Set the Inputs by dropping the column name 'genre'
X = music_data.drop(columns=["genre"])

# Set the Output by selecting only the 'genre'
Y = music_data["genre"]

# Initialize the Decision Tree Classifier
model = DTC()

# Train the model
model.fit(X, Y)
```

```
# Save the trained model to a file
joblib.dump(model, "music_recommender.pkl")  # Save as a .pkl file for best practice

# Optional: Predict a new value (uncomment to use)
# predictions = model.predict([[21, 1]])
```

Load the Data for predictions

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier as DTC
import joblib  # Import joblib directly

model = joblib.load("music_recommender.pkl")  # Save as a .pkl file for best practice
predictions = model.predict([[21, 1]])
predictions
```

Visualizing Data

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier as DTC
import joblib  # Import joblib directly
from sklearn import tree as T


# Read the CSV file
music_data = pd.read_csv("music.csv")

# Set the Inputs by dropping the column name 'genre'
X = music_data.drop(columns=["genre"])

# Set the Output by selecting only the 'genre'
Y = music_data["genre"]

# Initialize the Decision Tree Classifier
model = DTC()

# Train the model
model.fit(X, Y)


# Export To A Graph
T.export_graphviz(
                  model,
```

```
 out_file="music_recommender.dot",
 feature_names = ['age', 'gender'],
 class_names = sorted( Y.unique() ),
 label = 'all',
 rounded = True,
 filled=True


)
```