Master Boilerplate & Cheat Sheet for Data Science / ML Practicals

1. Common Libraries to Import

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

from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
mean_squared_error, r2_score

from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.cluster import KMeans, DBSCAN

from mlxtend.frequent_patterns import apriori, association_rules

from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller

from textblob import TextBlob
from nltk.corpus import stopwords
import re

from wordcloud import WordCloud
```

2. Data Loading + Basic Analysis

```
df = pd.read_csv("your_dataset.csv")
print(df.head())
print(df.info())
print(df.describe())
```

3. Missing Values + Duplicates Check

```
print(df.isnull().sum())
df = df.dropna(subset=['important_column'])
print("Duplicate rows:", df.duplicated().sum())
df = df.drop_duplicates()
```

Master Boilerplate & Cheat Sheet for Data Science / ML Practicals

4. Feature Scaling & Encoding

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df[['col1', 'col2']])

le = LabelEncoder()
df['encoded_column'] = le.fit_transform(df['category_column'])
```

5. Train-Test Split

```
X = df[['feature1', 'feature2']]
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

6. Linear / Logistic Regression

```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("RMSE:", mean_squared_error(y_test, y_pred, squared=False))
print("R²:", r2_score(y_test, y_pred))

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

7. Classification Comparison

```
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "Naive Bayes": GaussianNB(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

```
    print(f"\n{name}:")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
```

8. K-Means / DBSCAN Clustering

```
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X_scaled)
sns.scatterplot(x=X_scaled[:,0], y=X_scaled[:,1], hue=labels)

dbscan = DBSCAN(eps=0.5, min_samples=5)
labels = dbscan.fit_predict(X_scaled)
sns.scatterplot(x=X_scaled[:,0], y=X_scaled[:,1], hue=labels)
```

9. Apriori + Association Rules

```
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

10. Time Series Forecasting (ARIMA)

```
result = adfuller(df['Sales'])
print("ADF p-value:", result[1])

model = ARIMA(df['Sales'], order=(1,1,1))
model_fit = model.fit()
forecast = model_fit.forecast(steps=12)
forecast.plot()
```

11. Sentiment Analysis (TextBlob)

```
df['clean_text'] = df['ReviewText'].str.lower().apply(lambda x: re.sub(r'[^\w\s]', '', x))
df['sentiment'] = df['clean_text'].apply(lambda x: TextBlob(x).sentiment.polarity)
```

12. WordCloud

```
text = " ".join(df['clean_text'])
```

```
wordcloud = WordCloud(background_color='white').generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

13. Visualization Shortcuts

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

df['Month'] = pd.to_datetime(df['InvoiceDate']).dt.to_period('M')
monthly_sales = df.groupby('Month')['TotalPrice'].sum()
monthly_sales.plot(kind='line')
```

14. PySpark Sample

```
rdd = sc.textFile("file.txt").flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda a,b: a+b)
rdd.collect()

df.groupBy("department").agg(avg("salary")).show()
```