Python Data Science Cheat Sheet – Pandas, NumPy, and Scikit-Learn

NumPy – Numerical Computing

NumPy Basics

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])  # Creating an array
print(arr.shape)  # Shape of the array
print(arr.dtype)  # Data type of elements
```

NumPy Array Operations

```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

print(arr1 + arr2)  # Element-wise addition
print(arr1 * arr2)  # Element-wise multiplication
print(np.mean(arr1))  # Mean
print(np.std(arr1))  # Standard deviation
print(np.median(arr1))  # Median
```

NumPy Indexing & Slicing

```
arr = np.array([10, 20, 30, 40, 50])
print(arr[1:4])  # [20 30 40]
print(arr[::-1])  # Reverse array
```

NumPy Matrix Operations

```
matrix = np.array([[1, 2], [3, 4]])
print(matrix.T)  # Transpose
print(np.linalg.inv(matrix))  # Inverse of matrix
print(np.dot(matrix, matrix))  # Matrix multiplication
```

Random Numbers with NumPy

```
rand_arr = np.random.rand(3, 3)  # 3x3 matrix with random values
rand_ints = np.random.randint(0, 100, (3, 3))  # Random integers from 0-100
```

Pandas – Data Handling

Creating DataFrames

```
import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Hannah', 'Ian', 'Jack',
             'Kevin', 'Laura', 'Mona', 'Nathan', 'Olivia', 'Paul', 'Quincy', 'Rachel', 'Steve', 'Tom'],
    'Age': np.random.randint(20, 50, 20),
    'Salary': np.random.randint(30000, 100000, 20),
    'Department': np.random.choice(['HR', 'IT', 'Finance', 'Marketing'], 20)
}
df = pd.DataFrame(data)
print(df)
```

Reading & Writing Data

```
df.to_csv('output.csv', index=False)  # Save to CSV
df.to_excel('output.xlsx')  # Save to Excel
```

Data Selection & Filtering

```
print(df.head())  # First 5 rows
print(df.tail())  # Last 5 rows
print(df['Name'])  # Select a column
print(df.iloc[0])  # Select first row
filtered_df = df[df['Age'] > 30]  # Filter rows
print(filtered_df)
```

Handling Missing Values

```
df.loc[5, 'Salary'] = np.nan  # Introduce NaN value
df.fillna(0, inplace=True)  # Replace NaN with 0
df.dropna(inplace=True)  # Drop rows with NaN
df['Salary'].fillna(df['Salary'].mean(), inplace=True)  # Fill with mean
print(df)
```

Grouping & Aggregation

```
print(df.groupby('Department').mean())  # Group by Department and average
```

Merging & Joining

```
df1 = df[['Name', 'Age']]
df2 = df[['Name', 'Salary']]
merged_df = pd.merge(df1, df2, on='Name', how='inner')
print(merged_df)
```

Pivot Tables & Crosstabs

```
pivot_table = df.pivot_table(values='Salary', index='Department', aggfunc='mean')
crosstab = pd.crosstab(df['Department'], df['Age'])
print(pivot_table)
print(crosstab)
```

Scikit-Learn – Machine Learning

Data Preprocessing

```
from sklearn.preprocessing import StandardScaler, LabelEncoder

scaler = StandardScaler()
df[['Age', 'Salary']] = scaler.fit_transform(df[['Age', 'Salary']])
print(df)

encoder = LabelEncoder()
df['Department'] = encoder.fit_transform(df['Department'])
print(df)
```

Train-Test Split

```
from sklearn.model_selection import train_test_split

X = df[['Age', 'Salary']]
y = df['Department']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape)
```

Linear Regression

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(predictions)
```

Classification – Logistic Regression

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(y_pred)
```

Decision Trees & Random Forest

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

dt = DecisionTreeClassifier()
rf = RandomForestClassifier(n_estimators=100)

dt.fit(X_train, y_train)
rf.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)
y_pred_rf = rf.predict(X_test)
print(y_pred_rf)
```

Model Evaluation

```
from sklearn.metrics import accuracy_score, classification_report

print(accuracy_score(y_test, y_pred_rf))  # Model accuracy
print(classification_report(y_test, y_pred_rf))  # Classification report
```

Feature Selection

```
from sklearn.feature_selection import SelectKBest, f_classif

selector = SelectKBest(score_func=f_classif, k=2)
X_new = selector.fit_transform(X, y)
print(X_new)
```