# Data Analysis Cheat Sheet

## Objectives

* Summarize the main characteristics of the data.
* Gain a deeper understanding of the dataset.
* Uncover relationships between variables.
* Extract significant variables.

## Key Modules

* **Scientific Libraries:** NumPy, pandas, SciPy
* **Visualization Libraries:** Matplotlib, Seaborn
* **Algorithmic Libraries:** Scikit-learn, Statsmodels

## DataFrame Operations

### Finding Indexes

print(df.index[df['XYZ'] == my\_value].tolist())

## Database API

### Basic Operations

* **Connect:** Establish a connection with the database.
* **Cursor:** Create a cursor to run queries.
* **Commit:** Commit any pending transactions.
* **Rollback:** Roll back to the start of any pending transaction.
* **Close:** Close the connection to the database.

### Data Export Formats

Tables can be exported to:

* JSON
* CSV
* SQL
* Excel

## Data Analysis Steps

1. **Define the Business Problem:** Understand the data and determine the appropriate analysis.
2. **Import Data and Explore:**
   * Check data types: df.dtypes. Convert using df.astype(int).
   * Check for missing values: df.isnull().sum(). Replace with df.replace('?', np.NaN) and use df.info() for details.
   * Handle missing data:
     + Drop rows: df.dropna(axis=0)
     + Drop specific column entries: df.dropna(subset=["price"], axis=0, inplace=True)
     + Drop columns with excessive missing values: df.dropna(axis=1, thresh=0.75 \* len(df))
   * Reset index: df.reset\_index(drop=True, inplace=True)
   * Replace missing values with averages or frequency-based measures.

### Data Cleaning

* Rename columns: df.rename(columns={orig:new}, inplace=True)
* Strip whitespace:

data[data.columns] = data[data.columns].apply(lambda x: x.str.strip())

### Statistical Summary

* Basic statistics: df.describe(); for all columns, use df.describe(include='all').
* Value counts: df['column'].value\_counts().
* Unique values: df['column'].unique() and df['column'].nunique().
* Create box plots to visualize outliers and distributions.

## Relationship Analysis

* **Scatter Plot:**

sns.regplot(x='engine', y='price', data=df)

* **Scatter Matrix:**

pd.plotting.scatter\_matrix(df, alpha=0.2)

* **Heatmap:**

plt.pcolor(df, cmap='RdBu')

## Correlation Analysis

### Pearson Correlation

from scipy import stats

stats.pearsonr(df['engine'], df['price'])

### Chi-Square Test

Create a contingency table using pd.crosstab(), and perform the test:

from scipy.stats import chi2\_contingency

chi2\_contingency(contingency\_table, correction=True)

### ANOVA

To test significant differences between means:

from scipy import stats

stats.f\_oneway(grouped\_test2.get\_group('fwd')['price'],

grouped\_test2.get\_group('rwd')['price'],

grouped\_test2.get\_group('4wd')['price'])

## Binning and Grouping

Use np.linspace() for binning:

bins = np.linspace(min(df['price']), max(df['price']), no\_of\_bins)

group\_names = ['Low', 'Medium', 'High']

df['price-binned'] = pd.cut(df['price'], bins, labels=group\_names, include\_lowest=True)

### Grouping Data

Create summaries using groupby() and pivot tables.

## Normalization Techniques

* **Simple Scaling:** x / max
* **Min-Max Scaling:** (x - min) / (max - min)
* **Z-Score Normalization:** (x - mean) / std

## Categorical Variables

One-hot encoding:

pd.get\_dummies(df['column'])

## Date/Time Analysis

Convert to datetime:

df['mydatetimecolumn'] = pd.to\_datetime(df['mydatetimecolumn'])

df.set\_index('mydatetimecolumn', inplace=True)

### Aggregating Data

Resampling:

df.resample('H').sum()

## Modeling

**Linear and Polynomial Regression:** Use GridSearch for parameter tuning.

### Residual Analysis

Check for patterns using residual plots:

sns.residplot(x=df['highway-mpg'], y=df['price'])

### Evaluation Metrics

* **Mean Squared Error (MSE):**
* **R-Squared (R²):** Evaluate model performance.

### Cross-Validation

Use cross\_val\_score() for model validation.