Data Science Cheat Sheet

Table of Contents

---

1. Introduction to Data Science

What is Data Science?

- Data Science is the field of study that combines various disciplines such as statistics, machine learning, and data analysis to extract meaningful insights from structured and unstructured data.

Key Stages in a Data Science Project:

1. **Data Collection**: Gathering data from different sources.
2. **Data Preprocessing**: Cleaning and preparing the data for analysis.
3. **Exploratory Data Analysis (EDA)**: Exploring and visualizing data to understand patterns.
4. **Modeling**: Applying statistical and machine learning models to make predictions.
5. **Evaluation**: Assessing the performance of the model.
6. **Deployment**: Deploying the model for real-world use.

---

2. Data Preprocessing

Common Data Preprocessing Steps:

1. **Handling Missing Data**:
   - **Drop rows/columns** with missing values using `dropna()`.
   - **Impute missing values** using mean, median, or mode: `SimpleImputer()` in `sklearn`.
2. **Data Encoding**:
   - **One-hot encoding** for categorical variables: `pd.get_dummies()`.
   - **Label encoding** for ordinal variables: `LabelEncoder()` in `sklearn`.
3. **Scaling and Normalization**:
   - **Standardization**: `StandardScaler()` (mean=0, variance=1).

- **Min-Max Scaling**: `MinMaxScaler()` (range [0, 1]).

4. **Feature Selection**:
   - Removing irrelevant or highly correlated features using correlation matrix or model-based methods.
   - **Recursive Feature Elimination (RFE)**: `RFE()` in `sklearn`.
5. **Data Splitting**:
   - Split data into training and test sets using `train_test_split()` from `sklearn`.

---

3. Exploratory Data Analysis (EDA)

EDA Goals:

- Understand the distribution of data.
- Identify relationships between features.
- Detect outliers and anomalies.
- Identify missing or corrupt data.

Tools and Techniques:

1. **Descriptive Statistics**:
   - Mean, Median, Mode: `data.mean()`, `data.median()`, `data.mode()`.
   - Variance, Standard Deviation: `data.var()`, `data.std()`.
   - Skewness and Kurtosis: `data.skew()`, `data.kurt()`.
2. **Visualization Techniques**:
   - **Histograms**: `plt.hist()`.
   - **Boxplots**: `sns.boxplot()`.
   - **Pairplots**: `sns.pairplot()`.
   - **Correlation Matrix**: `sns.heatmap(data.corr())`.
   - **Scatter Plots**: `plt.scatter()`.
   - **Bar Plots**: `sns.barplot()`.

---

4. Statistical Analysis

Basic Statistical Tests:

1. **Hypothesis Testing**:
   - **t-test**: Used to compare means between two groups.
   - **ANOVA**: Used to compare means across multiple groups.
   - **Chi-Square Test**: Tests for independence between categorical variables.
2. **Confidence Intervals**:
   - A range of values likely to contain the population parameter with a given level of confidence.

3. **P-value**:
    - Measures the probability that the observed result is due to chance. A small p-value (typically < 0.05) indicates statistical significance.

---

5. Machine Learning Algorithms

Supervised Learning:

1. **Linear Regression**: For predicting continuous values.
    - `LinearRegression()` in `sklearn`.
2. **Logistic Regression**: For binary classification problems.
    - `LogisticRegression()` in `sklearn`.
3. **Decision Trees**: For classification and regression tasks.
    - `DecisionTreeClassifier()`, `DecisionTreeRegressor()` in `sklearn`.
4. **Random Forest**: Ensemble method using multiple decision trees.
    - `RandomForestClassifier()`, `RandomForestRegressor()` in `sklearn`.
5. **Support Vector Machine (SVM)**: For classification tasks, can also be used for regression.
    - `SVC()` in `sklearn`.
6. **K-Nearest Neighbors (KNN)**: For classification and regression.
    - `KNeighborsClassifier()`, `KNeighborsRegressor()` in `sklearn`.
7. **Naive Bayes**: Based on Bayes' Theorem, used for classification.
    - `GaussianNB()`, `MultinomialNB()` in `sklearn`.

Unsupervised Learning:

1. **K-Means Clustering**: For grouping data into clusters.
    - `KMeans()` in `sklearn`.
2. **Hierarchical Clustering**: For creating a tree of clusters.
    - `AgglomerativeClustering()` in `sklearn`.
3. **Principal Component Analysis (PCA)**: Dimensionality reduction technique.
    - `PCA()` in `sklearn`.
4. **DBSCAN**: Density-based spatial clustering.
    - `DBSCAN()` in `sklearn`.

---

6. Model Evaluation

Evaluation Metrics for Classification:

1. **Accuracy**: Proportion of correct predictions.

- `accuracy_score()` in `sklearn`.

2. **Precision, Recall, and F1-score**: Metrics for evaluating classification performance.
   - `precision_score()`, `recall_score()`, `f1_score()` in `sklearn`.
3. **Confusion Matrix**: Displays true positives, false positives, true negatives, and false negatives.
   - `confusion_matrix()` in `sklearn`.
4. **ROC Curve and AUC**: For binary classification problems.
   - `roc_curve()` and `auc()` in `sklearn`.

Evaluation Metrics for Regression:

1. **Mean Absolute Error (MAE)**: Average of the absolute errors.
   - `mean_absolute_error()` in `sklearn`.
2. **Mean Squared Error (MSE)**: Average of the squared errors.
   - `mean_squared_error()` in `sklearn`.
3. **R-squared**: Represents the proportion of variance explained by the model.
   - `r2_score()` in `sklearn`.

---

7. Data Visualization

Key Libraries:

1. **Matplotlib**:
   - Basic plotting library: `plt.plot()`, `plt.scatter()`, `plt.hist()`.
2. **Seaborn**:
   - Built on top of Matplotlib, provides more advanced plots like heatmaps, boxplots, and pairplots.
   - `sns.heatmap()`, `sns.boxplot()`, `sns.barplot()`.
3. **Plotly**:
   - Interactive plotting library for web-based visualization.
   - `plotly.express` for easy-to-use interactive charts.
4. **Altair**:
   - Declarative statistical visualization library.
   - `alt.Chart()` for creating interactive plots.

---

8. Deep Learning

Key Concepts in Deep Learning:

1. **Neural Networks**: Composed of layers of nodes (neurons) that learn from data.
2. **Convolutional Neural Networks (CNN)**: Specialized for image processing tasks.

3. **Recurrent Neural Networks (RNN)**: Suitable for sequential data (e.g., time series, text).

Deep Learning Libraries:

1. **TensorFlow**: Popular library for creating deep learning models.
   - `tensorflow.keras` for building models.
2. **PyTorch**: Another widely used deep learning framework.
   - `torch.nn` for building neural networks.
3. **Keras**: A high-level API for building deep learning models (now part of TensorFlow).

---

9. Libraries & Frameworks

1. **NumPy**: Core library for numerical computing, array manipulation.
   - `np.array()`, `np.mean()`, `np.std()`.
2. **Pandas**: For data manipulation and analysis.
   - `pd.DataFrame()`, `pd.read_csv()`, `df.head()`.
3. **Matplotlib**: For creating static, animated, and interactive plots.
   - `plt.plot()`, `plt.scatter()`, `plt.bar()`.
4. **Scikit-learn**: Machine learning library with a wide range of algorithms and tools for model selection, evaluation, and preprocessing.
   - `sklearn.model_selection`, `sklearn.preprocessing`.
5. **Statsmodels**: For statistical models and tests.
   - `sm.OLS()`, `sm.tTest()`.

---

10. Best Practices and Tips

1. **Data Cleaning**: Ensure data is free of inconsistencies, missing values, and outliers.
2. **Feature Engineering**: Create new features based on domain knowledge to improve model performance.
3. **Cross-Validation**: Use techniques like k-fold cross-validation to validate model performance.
4. **Hyperparameter Tuning**: Use grid search or random search to find the best hyperparameters for your model.
5. **Model Selection**: Choose models based on the problem type (classification, regression, etc.) and dataset characteristics.