Useful commands for pandas.DataFrame

1. Take a quick look

- df.head(10)	: return the first 10 rows of the dataframe
- df.tail(10)	: return the last 10 rows of the dataframe
- df.shape	: return the dimensions of the dataframe
- df.info()	: return a summary of columns, no. of non-null value & data type
- df[ColA] .value_counts	: return counts of unique values in ColA
- df.describe	: return some statistics for df / df subsets

2. Data Manipulation

- **Drop named columns**
  - ➢ df.drop(columns = [[ColA, ColB, …]], inplace = True)
- **Split concatenated columns**
  - ➢ df[[cola, colB]] = df[concat_col].str.split(‘,’, expand = True)
- **Sort the order**
  - ➢ df[ColA].sort_values(ascending = False)
- **Groupby**
  - ➢ df.groupby(ColA)…some more operations here…

Useful commands for matplotlib.pyplot as plt

1. Histogram Graph

```
plt.hist(
        pd.DataFrame[ColA]
)
plt.xlabel( … )
plt.ylabel ( … )
plt.title( … )
```

2. Boxplot Graph

```
plt.box(
        pd.DataFrame[ColA],
)
plt.xlabel( … )
plt.title( … )
```

Useful commands for plotly.express as px

1. Scatter Map Graph

```
fig = px.scatter_map(
        pd.DataFrame,
        lat = colA,
        lon = colB,
        center={"lat": … , "lon": … },
        width=600,
        height=600,
        hover_data=[ColC]
)
fig.update_layout(mapbox_style="open-street-map")
fig.show()
```

Popular Models (Regression)

1. **Linear Regression** (sklearn.linear_model.LinearRegression)

   Ordinary least squares Linear Regression

   LinearRegression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares
   - coef_
   - rank_
   - intercept_
2. **Auto Regressive** (statsmodels.tsa.ar_model.AutoReg)
   - cooperate with PACF to study the correlation of previous values.
3. **ARMA** (statsmodels.tsa.ar_model.ARIMA)

Popular Models (Classification)

1. **Logistic Regression** (sklearn.linear_model.LogisticRegression)
   - max_iter

     handling overfitting / underfitting by changing the number of gradient descents to run and tune the parameters
2. **Decision Tree** (sklearn.tree.DecisionTreeClassifier)
   - max_depth
3. **Random Forest** (sklearn.ensemble.RandomForestClassifier)

   Random Forest Classifier fits a number of decision tree trained by different subsets of data
   - n_estimators: the number of trees

Preprocessing

1. **SimpleImputer** (sklearn.impute.SimpleImputer)

   Replace missing values with specific value, such as mean, median
2. **OrdinalEncoder** (sklearn.preprocessing.OrdinalEncoder)

   Encode categorical features with integer array (0…n-1), applicable for values with ordering.
3. **OneHotEncoder** (sklearn.preprocessing.OneHotEncoder)

   Encode categorical features with binary format, applicable for values without ordering.

Random Sampling

1. **Oversampling** (imblearn.over_sampling.RandomOverSampler)
   Randomly duplicate records from the minority class
2. **Undersampling** (imblearn.under_sampling.RandomUnderSampler)
   Randomly remove records from the majority class

Cross-Validation

By resampling and training different models, CV allows to check actual performance during testing and deployment.

1. **sklearn.model_selection.cross_val_score** (pipeline, x_train, y_train, cv=5, n_jobs=-1)
   *5-fold is a usual practice for cross-validation
   *n_jobs refers to the number of parallel jobs while -1 means using all processors

2. **sklearn.model_selection.GridSearchCV** (pipeline, param_grid, cv=5, n_jobs=-1, verbose= 1/2/3)
   *param_grid refers to dictionary of all hyper-parameters for picking the best model
   *verbose refers the amount of information shown

MongoDB (NoSQL)

Running as a non-relational database, and it can handle storage for structured, semi-structured & unstructured data.

4. Structure: Database → Collection (= table) → Document (= record)

Useful read commands:

1. *List(client.list()_databases()) / List(database.list_collections())*
2. *db.collection.find/findOne(<query>, <projection>, <options>)* **[Ref]**
3. *collection.aggregate([{ … }])* **[Ref]**

SQLite (SQL)

Running as SQL database (similar syntax with PostgreSQL), which is a relational database, with ACID guarantees, preferably working with but not limited structured data.

5. Structure: SQLite schema → table

Useful commands:

1. Aggregate functions: count(), distinct(), avg(), sum()
2. Queries:
   - AS – to rename columns & tables (create alias)
   - WHERE – to filter out records fitting the selection

- JOIN – to merge the columns in tables
- LIMIT – to output the specific number of results

SQLite + Pandas:

1. sqlite3.connect($PATH_TO_SQLite) return a *Connection Object*
2. pandas.read_sql($YOUR_QUERY, connection) return a *DataFrame Object*