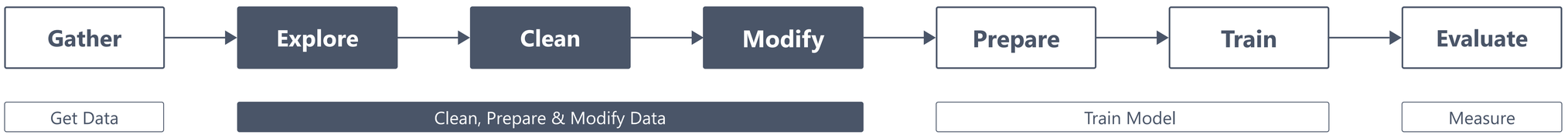
AI Workflow

Data Exploration, Cleaning and Modification



Note: Pandas is being utilized here due to its excellent compatibility with Spark, making it suited for Big Data Processing on a scalable system (assisted through the use of koalas) (=data engineering)
Once data is cleansed, SciKit can be utilized for model creation (= data science)

EXPLORE

EDA: Exploratory Data Analysis

TYPE

https://docs.python.org/3/library/functions.html#type

With one argument, return the type of an object

Command: type(var_name)

KEYS

https://docs.python.org/3/library/stdtypes.html?highlight=keys#dict.keys

Return a new view of the dictionary's keys

Command: var_name.keys()

NP & DF SHAPE

https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html

Tuple of dimensions for DataFrame or Numpy Array

Command: var_name.shape

DF HEAD

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.head.html

Return the first n rows.

Command: df = df.head()

Example

```
> df = pd.DataFrame({'animal':['alligator', 'bee',
'falcon', 'lion', 'monkey', 'parrot', 'shark', 'whale', 'zebra']})
> df

      animal
0  alligator
1        bee
2     falcon
3       lion
4     monkey
```

PD MATRIX

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.plotting.scatter_matrix.html

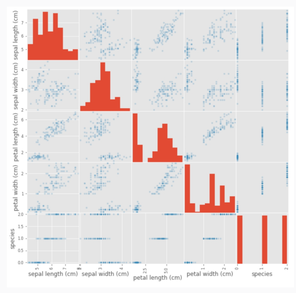
Draw a matrix of scatter plots.

Command: pd.scatter_matrix()

Example

```
# Load some data
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris['data'],
columns=iris['feature_names'])
iris_df['species'] = iris['target']

pd.scatter_matrix(iris_df,
alpha=0.2, figsize=(10, 10))
plt.show()
```



PD DESCRIBE

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

Generate descriptive statistics that summarize the central

Command: df = df.describe()

Example

```
> s = pd.Series([1, 2, 3])
> s.describe()
count    3.0
mean     2.0
std      1.0
min      1.0
25%      1.5
50%      2.0
75%      2.5
max      3.0
dtype: float64
```

```
> s = pd.Series(['a', 'a', 'b', 'c'])
> s.describe()
count     4
unique    3
top       a
freq      2
dtype: object
```

CLEAN

Note: import pandas as pd & import numpy as np

DROP NULLS

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html

Return object with labels on given axis omitted where alternately any or all of the data are missing

Command: df = df.dropna()

Example

```
> df = pd.DataFrame([[np.nan, 1], [2, 3]])
> df.dropna()
0 [2, 3]
```

IMPUTE

https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html

Instead of discarding data, it's better to "Impute" it, i.e., to infer it from known part of the data

Example

```
> df = pd.DataFrame(np.random.randn(5, 3),
index=['a', 'b', 'c', 'd', 'e'], columns=['one', 'two', 'three'])

> df['one']['a'] = None # Set a value as undefined
> df = df.fillna(0) # Fill missing values with 0
> df = df.fillna(df.median()) # Fill missing values with median
```

REPLACE

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.replace.html

Replace values by another one (e.g. NaN values by 0)

Command: s.replace(search_value, new_value)

Example

```
> df = pd.DataFrame([0])
> s.replace(0, 10) # Replace 0 by 10
   0
0 10
```

STANDARIZE (σ)

https://en.wikipedia.org/wiki/Standard_score

Standardization or z-score normalization takes into account the standard deviation
Formula: z = (x - μ) / σ where μ = mean σ = standard deviation

Example

```
> df = pd.DataFrame(np.random.randn(5, 3),
index=['a', 'b', 'c', 'd', 'e'], columns=['one', 'two', 'three'])

> df = (df - df.min()) / (df.max() - df.min())
```

NORMALIZE

https://en.wikipedia.org/wiki/Feature_scaling

Rescale the data to have values between 0 and 1
Formula: z = (x - min(x)) / (max(x) - min(x))

Example

```
> df = pd.DataFrame(np.random.randn(5, 3),
index=['a', 'b', 'c', 'd', 'e'], columns=['one', 'two', 'three'])

> df = (df - df.min()) / (df.max() - df.min())
```

MODIFICATION

BINNING

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.cut.html

Return object with labels on given axis omitted where alternately any or all of the data are missing

Example

```
> rand_list = [np.random.randint(0, 100) for i in range(50)]
> pd.cut(rand_list, 5) # Create 5 equal sized bins
```

ONE-HOT ENCODING

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

Convert categorical features to a numerical array of 0 or 1. E.g. ['warm'] for [hot, warm, cold] becomes [0, 1, 0]

Example

```
> data = pd.DataFrame([['Male', 1], ['Female', 3], ['Female', 2]])

# One Hot encode and prefix new columns with ohe_
> pd.get_dummies(df, prefix='ohe')
```

LABEL ENCODING

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.astype.html

Converts categorical feature to a numerical array of numbers E.g. ['warm'] for [hot, warm, cold] becomes [1]

Example

```
> df = pd.DataFrame([['Male', 1], ['Female', 3], ['Female', 2]])
> df[0] = df[0].astype('category') # Convert to category
> df['0_encoded'] = df[0].cat.codes # Label encoding: .cat.codes
```

DATE EXTRACTION

https://docs.python.org/3/library/datetime.html

Extract parts of the date

Example

```
> from datetime import date
> df = pd.DataFrame({ 'date': ['01-01-2000', '31-12-2019'] })
> df['date'] = pd.to_datetime(df.date, format="%d-%m-%Y")
> df['date'].dt.year # Print year
> df['date'].dt.day_name() # Print weekday
```

PIVOT TABLES

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.pivot_table.html

A table of statistics that summarize the data of a more extensive table. E.g. list of countries and cities to count of cities in that country

Example

```
> df = pd.DataFrame({ "city": ["brussels", "antwerp", "gent",
"seattle"], "country": ["BE", "BE", "BE", "USA"] })
> pd.pivot_table(df, values="city", index=['country'],
    aggfunc=np.count_nonzero)
```

SPLITTING

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.split.html

Split the characters in a column. E.g. name to first_name and last_name is quite common.

Example

```
> df = pd.DataFrame({ "name": ["Xavier Geeringck",
      "Jane Middle Doe"] })
> df['first_name'] = df.name.str.split(" ").map(lambda x: x[0])
> df['last_name'] = df.name.str.split(" ")
      .map(lambda x: "   ".join(x[1:]))
```