Data Preprocessing

Importing Data

df.to_csv(filename) # Writes to a CSV file

df.to_excel(filename) # Writes to an Excel file

df.to_sql(table_name, connection_object) # Writes to a SQL table

df.to_json(filename) # Writes to a file in JSON format

df.to_html(filename) # Saves as an HTML table

df.to_clipboard() # Writes to the clipboard

Exploring Data

df.shape() # Prints number of rows and columns in Dataframe

df.head(n) # Prints first n rows of the DataFrame

df.tail(n) # Prints last n rows of the DataFrame

df.info() # Index, Datatype and Memory information

df.describe() # Summary statistics for numerical columns

s.value_counts(dropna=False) # Views unique values and counts

df.apply(pd.Series.value_counts) # Unique values and counts for all columns

df.describe() # Summary statistics for numerical columns

df.mean() # Returns the mean of all columns

df.corr() # Returns the correlation between columns in a DataFrame

df.count() # Returns the number of non-null values in each DataFrame column

df.max() # Returns the highest value in each column

df.min() # Returns the lowest value in each column

df.median() # Returns the median of each column

df.std() # Returns the standard deviation of each column

Data Selecting

df[col] # Returns column with label col as Series

df[[col1, col2]] # Returns Columns as a new DataFrame

s.iloc[0] # Selection by position (selects first element)

s.loc[0] # Selection by index (selects element at index 0) df.iloc[0,:] # First row

df.iloc[0,0] # First element of first column

df.iloc[:,:,-1].values # Extract the independent variables (Features)

df.iloc[:, -1].values # Extract the dependent variable (Target)

Data Cleaning

df.columns = ['a','b','c'] # Renames columns

pd.isnull() # Checks for null Values, Returns Boolean Array

pd.notnull() # Opposite of s.isnull()

df.dropna() # Drops all rows that contain null values

df.dropna(axis=1) # Drops all columns that contain null values

df.dropna(axis=1,thresh=n) # Drops all rows have have less than n non null values

df.fillna(x) # Replaces all null values with x

s.fillna(s.mean()) # Replaces all null values with the mean (mean can be replaced with almost any function from the statistics section)

s.astype(float) # Converts the datatype of the series to float

s.replace(1,'one') # Replaces all values equal to 1 with 'one'

s.replace([1,3],['one','three']) # Replaces all 1 with 'one' and 3 with 'three'

df.rename(columns=lambda x: x + 1) # Mass renaming of columns

df.rename(columns={'old_name': 'new_ name'}) # Selective renaming

df.set_index('column_one') # Changes the index

df.rename(index=lambda x: x + 1) # Mass renaming of index

Filter, Sort and Group By

df[df[col] > 0.5] # Rows where the col column is greater than 0.5

df[(df[col] > 0.5) & (df[col] < 0.7)] # Rows where 0.5 < col < 0.7

df.sort_values(col1) # Sorts values by col1 in ascending order

df.sort_values(col2,ascending=False) # Sorts values by col2 in descending order

df.sort_values([col1,col2], ascending=[True,False]) # Sorts values by col1 in ascending order then col2 in descending order

df.groupby(col) # Returns a groupby object for values from one column

df.groupby([col1,col2]) # Returns a groupby object values from multiple columns df.groupby(col1)[col2].mean() # Returns the mean of the values in col2, grouped by the values in col1 (mean can be replaced with almost any function from the statistics section)

df.pivot_table(index=col1, values= col2,col3], aggfunc=mean) # Creates a pivot table that groups by col1 and calculates the mean of col2 and col3

df.groupby(col1).agg(np.mean) # Finds the average across all columns for every unique column 1 group

df.apply(np.mean) # Applies a function across each column

df.apply(np.max, axis=1) # Applies a function across each row

Joining and Combining

df1.append(df2) # Adds the rows in df1 to the end of df2 (columns should be identical)

pd.concat([df1, df2],axis=1) # Adds the columns in df1 to the end of df2 (rows should be identical)

df1.join(df2,on=col1,how='inner') # SQL-style joins the columns in df1 with the columns on df2 where the rows for col have identical values. how can be one of 'left', 'right', 'outer', 'inner'

Writing Data

df.to_csv(filename) # Writes to a CSV file

df.to_excel(filename) # Writes to an Excel file

df.to_sql(table_name, connection_object) # Writes to a SQL table

df.to_json(filename) # Writes to a file in JSON format

df.to_html(filename) # Saves as an HTML table

df.to_clipboard() # Writes to the clipboard

Data Transformation and Splitting using sklearn Package

# Filling missing value with mean

from sklearn.preprocessing import Imputer

imputer= Imputer(missing_values ='NaN', strategy='mean', axis = 0)

imputerimputer= imputer.fit(df[:, 1:3])

df[:, 1:3]= imputer.transform(df[:, 1:3])

df

# Encoding a variable

from sklearn.preprocessing import LabelEncoder

label_encoder_df= LabelEncoder()

df[:, 0]= label_encoder_df.fit_transform(df[:, 0])

```
df
# Dummy encoding
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)

# Train Test split
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.2)
X_train, X_test, y_train, y_test = train_test_split(df[list_of_x_cols], df[y_col], test_size=0.2)
```