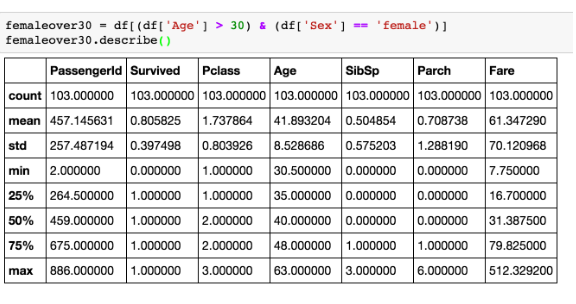
Big Data CourseWork

# Data Munging

1. Group data then fill it with the median - df['A'].fillna(df['A'].median())
2. Make sure numerical values are float64
3. 7 steps mastering data preparation - <https://www.kdnuggets.com/2017/06/7-steps-mastering-data-preparation-python.html>
4. 6 steps to wrangling – Discovering, structuring, cleaning, enriching, validating, publishing
5. Df.info() – shows how many entries and columns
6. Df.ifnull(df).any() another way of checking for missing values
7. Plot histograms, scatterplots, areaplots, line plot, KDE(kernel density estimation plot)
8. For exploratory data analysis its better to work with a subset of the data or filtering out parts of the data to match a certain criterion
9. 
10. Follow the eda steps for data exploration - <https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/>
11. Steps for data science in detail - <https://medium.com/@ODSC/top-data-wrangling-skills-required-for-data-scientists-8a6b7dc604a7>
12. Don’t replace anything within the testing set (Remove but do not change) , an evaluation on how the model performs. (training data) can cause data contamination.
13. Look at the csv file, clean all the NAN, clean out columns drop unnecessary columns, check all the types, check statistics df.describe(), check which ones should be numerical, check different datatypes (float /integer), don’t change any of the data, format the date correctly
14. Visualisation – check for meaningful patterns, have a look at dates that can be interpreted by an algorithm, missing values+outliers
15. Properties of a clean dataset
16. <https://github.com/rougier/matplotlib-tutorial>

# Predictive model

1. Use this website as a guide for building the predictive model - <https://www.analyticsvidhya.com/blog/2015/09/build-predictive-model-10-minutes-python/>
2. Predictive model is a process that uses the data and probability to forecast outcomes.
3. Different predictive models - <https://www.microstrategy.com/us/resources/introductory-guides/predictive-modeling-the-only-guide-you-need>
4. Predict sale price
5. Construct a linear model for sale price and justify why you chose that model
6. Justify your choice of predictors
7. Choose predictors and that will define my model

import statsmodels.formula.api as smf

def model(df):

df.rename(columns = {'SALE PRICE':'SALE\_PRICE'}, inplace= True)

df.rename(columns = {'GROSS SQUARE FEET':'GROSS\_SQUARE\_FEET'}, inplace= True)

model = smf.ols(formula='SALE\_PRICE ~ GROSS\_SQUARE\_FEET', data=df).fit()

print(model.params)

print('R Squared = {}'.format(model.rsquared))

print(model.summary())

model(df)

from sklearn import svm, feature\_selection, linear\_model

num\_cols = df.select\_dtypes(include=[np.number]).copy()

feature\_cols = df.columns.values.tolist()

feature\_cols.remove('SALE PRICE')

feature\_cols.remove('NEIGHBORHOOD')

feature\_cols.remove('TAX CLASS AT PRESENT')

feature\_cols.remove('BLOCK')

feature\_cols.remove('LOT')

feature\_cols.remove('ADDRESS')

feature\_cols.remove('ZIP CODE')

feature\_cols.remove('YEAR BUILT')

feature\_cols.remove('TAX CLASS AT TIME OF SALE')

feature\_cols.remove('SALE DATE')

feature\_cols.remove('BUILDING CLASS AT TIME OF SALE')

feature\_cols.remove('BUILDING CLASS CATEGORY')

feature\_cols.remove('BUILDING CLASS AT PRESENT')

feature\_cols.remove('YEAR SOLD')

#feature\_cols.remove('')

#feature\_cols.remove('')

XO =df[feature\_cols]

YO = df['SALE PRICE']

estimator = svm.SVR(kernel="linear")

selector = feature\_selection.RFE(estimator, 5, step=1)

selector = selector.fit(XO, YO)

select\_features = np.array(feature\_cols)[selector.ranking\_ == 1].tolist()

print(select\_features)

from sklearn.model\_selection import train\_test\_split

X = df[select\_features]

Y = df['SALE PRICE']

trainX, testX, trainY, testY = train\_test\_split(X, Y, test\_size=0.2)

lm = linear\_model.LinearRegression()

lm.fit(trainX, trainY)

# Inspect the calculated model equations

print("Y-axis intercept {:6.4f}".format(lm.intercept\_))

print("Weight coefficients:")

for feat, coef in zip(select\_features, lm.coef\_):

print(" {:>20}: {:6.4f}".format(feat, coef))

# The value of R^2

print("R squared for the training data is {:4.3f}".format(lm.score(trainX, trainY)))

print("Score against test data: {:4.3f}".format(lm.score(testX, testY)))