**Functions**

1. **Statistics basics in Python**
2. Mean

mean\_car\_speed = np.mean(car\_speeds)

1. Indexing numbers or strings

survey\_responses = ["none", "some", "a lot", "none", "a few", "none", "none"]

survey\_scale = ["none", "a few", "some", "a lot"]

survey\_numbers = [survey\_scale.index(response) for response in survey\_responses]

1. Skewness in data

from scipy.stats import skew

positive\_skew = skew(test\_scores\_positive)

negative\_skew = skew(test\_scores\_negative)

no\_skew = skew(test\_scores\_normal)

1. Kurtosis for outliers and distribution

from scipy.stats import kurtosis

kurt\_platy = kurtosis(test\_scores\_platy)

kurt\_lepto = kurtosis(test\_scores\_lepto)

kurt\_meso = kurtosis(test\_scores\_meso)

1. Median

median = numpy.median(test\_scores\_negative)

1. Standard deviation

std\_dev = nba\_stats["pf"].std()

1. Generating normal distribution

from scipy.stats import norm

points = np.arange(-1, 1, 0.01)

points1 = np.arange(-10, 10, 0.1)

probabilities = norm.pdf(points, 0, .3)

probabilities1 = norm.pdf(points1, 0, 2)

1. Pearson coefficient for correlation

from scipy.stats.stats import pearsonr

r, p\_value = pearsonr(nba\_stats["fga"], nba\_stats["pts"])

1. Covariance between variables

from numpy import cov

cov\_fta\_blk = cov(nba\_stats["fta"], nba\_stats["blk"])[0,1]

1. Linear Regression for variables and predicting others

from scipy.stats import linregress

slope, intercept, r\_value, p\_value, stderr\_slope = linregress(wine\_quality["density"], wine\_quality["quality"])

predicted\_y = numpy.asarray([slope \* x + intercept for x in wine\_quality["density"]])

residuals = (wine\_quality["quality"] - predicted\_y) \*\* 2

rss = sum(residuals)

1. Standard error

predicted\_y = np.asarray([slope \* x + intercept for x in wine\_quality["density"]])

residuals = (wine\_quality["quality"] - predicted\_y) \*\* 2

rss = sum(residuals)

s\_e = (rss/(len(wine\_quality["quality"])-2))\*\* (1/2)

1. Lowest value in the column

lowest\_income\_county = income["county"][income["median\_income"].idxmin()]

high\_pop = income[income["pop\_over\_25"] > 500000]

lowest\_income\_high\_pop\_county = high\_pop["county"][high\_pop["median\_income"].idxmin()]

1. Random number generator

# Returns a random integer between the numbers 0 and 10, inclusive.

num = random.randint(0, 10)

# Generate a sequence of 10 random numbers between the values of 0 and 10.

random.seed(10)

random\_sequence = [random.randint(0, 10) for \_ in range(10)]

1. Random sample from the list

shopping = [300, 200, 100, 600, 20]

random.seed(1)

shopping\_sample = random.sample(shopping, 4)

1. To choose columns with no missing data

train\_null\_counts = train.isnull().sum()

print(train\_null\_counts)

df\_no\_mv = train[train\_null\_counts[train\_null\_counts==0].index]

1. To select columns with text responses

text\_cols = df\_no\_mv.select\_dtypes(include=['object']).columns

1. To change text column into categorical variables

text\_cols = df\_no\_mv.select\_dtypes(include=['object']).columns

for col in text\_cols:

print(col+":", len(train[col].unique()))

train[col] = train[col].astype('category')

train['Utilities'].cat.codes.value\_counts()

1. To attach columns in dataframes

train = pd.concat([train, col\_dummies], axis=1)