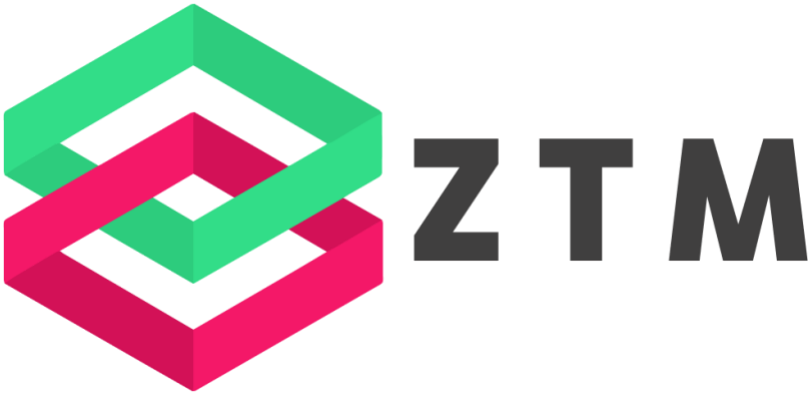

BUSINESS ANALYTICS w/ PYTHON CHEAT SHEET

DIOGO RESENDE

V1.01

HEEELLLOOOOO!

I'm Andrei Neagoie, Founder and Lead Instructor of the Zero To Mastery Academy.

After working as a Senior Software Developer over the years, I now dedicate 100% of my time to teaching others valuable software development skills, help them break into the tech industry, and advance their careers.

In only a few years, **over 750,000 students** around the world have taken Zero To Mastery courses and many of them are now working at top tier companies like Apple, Google, Amazon, Tesla, IBM, Facebook, and Shopify, just to name a few.

This cheat sheet, created by our Business Analytics instructor (Diogo Resende) provides you with the key Business Analytics definitions and concepts, and Python fundamentals, that you need to know and remember.

If you want to not only learn Business Analytics but also get the exact steps to build your own projects and get hired as a Data Scientist or Developer, then check out our Career Paths.

Happy Coding!
Andrei

Founder & Lead Instructor, Zero To Mastery
Andrei Neagoie

P.S. I also recently wrote a book called Principles For Programmers. You can download the first five chapters for free here.

Business Analytics Cheatsheet

Table of Contents

Statistics & Regression

Econometrics & Causal Inference

Segmentation

Definitions
RFM Python
Gaussian Mixture Model Python

Predictive Analytics

Definitions
Random Forest in Python
Facebook Prophet in Python

Statistics & Regression

Definitions

Statistics

The science of gathering, collecting, manipulating, analzying, interpreting and visualizing data.

Descriptive Analytics

Interpretation and visualization of data to determine trends and relationships.

Mean

The mean is the sum of all values divided by the number of occurrences; is the expected value we should get by picking randomly out of the sample. It should be used for continuous and symmetrical distributions.

Mode

The most frequent number or element in an ordered set. Should be used for qualitative-nominal variables.

Median

The 50th percentile or center of an ordered set. Robust to outliers. Should be used for asymmetrical and skewed distributions.

Correlation

Measures the strength of a relationship between 2 variables. The value varies between 1 and -1, with 1 being a strong positive correlation, -1 a strong negative correlation, and 0 no relationship.

Standard Deviation

Measures how dispersed is a data set. It should be asses conjointly with the mean. Rule of thumb: if the Standard Deviation is bigger than the mean, the data is dispersed.

Normal Distribution

Symmetric shape with the mean in the middle. The data is mainly around the mean values, meaning that it is not very dispersed, and its shape is similar to a bell. The Normal Distribution also brought the 68, 95, and 99 rule, representing the share of observations within one, two, and three standard deviations. Within 1 standard

deviation, there are 68% of the observations. Within 2 Standard Deviations, there are 95% of the observations. Finally, within 3 Standard Deviations, there is 99.7%.

P-value

A p-value varies between 0 and 1, and is thus a probability. The formal definition is the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuming the null hypothesis is correct.

Null Hypothesis (H0)

The hypothesis being tested for statistical significance.

Alternative Hypothesis (H1)

The statement being tested against the null hypothesis for statistical significance.

Shapiro-Wilk Test

A statistical test for normal distributions. The null hypothesis is that the distribution is Normal. Hence, if the p-value is above 0.05, we do not reject the null hypothesis

Standard Error of the Mean

Estimates how far the sample mean is likely to be from the population mean. Different from Standard Deviation. Whereas the standard deviation measures how much the observations differ from the mean, the standard error compares the sample's mean versus the population's mean.

Z Score

A standardization score, also known as the z-score.Provides an idea of how far away is a data point from its mean. Another way to look at it is that the z-score compares the observation values to a normal distribution.

Confidence Interval

How much the parameter or coefficient can vary around its mean for a certain confidence level (80%, 90%, 95%, etc…). From a methodological perspective, the Confidence Interval is a combination of the mean, the z score or confidence level, and the standard error of the mean.

T-test

Ttest any statistical hypothesis that the test statistic follows a Student's t-distribution under the null hypothesis. There are two essential things we need to understand—first,

the null hypothesis. We need to have one. The second part to digest is “test statistic follows Student’s t-distribution.” Simply, the variables are continuous.

Chi-square test

Determines whether there is a statistically significant difference between the expected frequencies and the observed frequencies of categorical variables. Tests whether there is a relationship among 2 categorical variables.

Linear Regression

The study of the relationship between one output or independent variable (continuous) and at least one independent variable.

R squared

Measures how much variability the model explains. For when you have only one Independent Variable

Adjusted R squared

Measures how much variability the model explains. For when you have more than one Independent Variable.

Dummy Variable

Type of variable that has binary possibilities (0 or 1)

Dummy Variable Trap

When you have perfect multicollinearity between your dummy variables. You should always omit one dummy variable to prevent it.

Multilinear Regression

A Regression model with more than 1 Independent Variable

Outliers

Extreme, isolated and uncommon values in a sample. Can potentially hurt your analysis.

Underfitting model

When a model cannot explain any instance

Overfitting model

When a model explains every instance, resulting in not working in unseen data

Training set

A random majority sample of the data used to train the model.

Test set

A random minority sample, or what is not used for the training data, to test the model.

Accuracy

The degree to which the model is correct on average

Mean Absolute Error (MAE)

KPI to measure the accuracy. Measure the absolute error between the prediction and actual value. Interpretable

Root Squared Mean Error (RSME)

KPI the to measure the accuracy. Not interpretable. Penalizes errors in extreme values.

Logistic Regression

The study of the relationship between one output or independent variable (Binary) and at least one independent variable.

Confusion Matrix

Cross Table that summaries the accuracy of Classification Models like the logistic Regression. Has four possibilities: True Negatives, True Positives

True Negative

When the model predicted a false or zero value and was correct.

True Positive

When the model predicted a true or one value that was correct.

- *False Negative **

When the model predicted a False value that was in fact, true.

False Positive

When the model predicted a True Value when the actual value is False.

F1 Score

Accuracy measure for unbalanced classification data sets - when the frequency of either of the dependent variables possibilities (0 or 1) is less than 30%, which makes the other

possibility more than 70%.

Key Libraries for Statistics and Descriptive Analytics in Python

```
import pandas as pd
import seaborn as sns
import scipy.stats as st
import math as m
import statsmodels.stats.api as sm
```

Import a Data Set

```
df = pd.read_csv("filename.csv")
df.head()
```

Compute Mean, Mode and Median

```
numbers = pd.Series([10, 8, 9, 1, 8, 0, 100])
numbers.mean() # 19.4
numbers.mode() # 9.0
numbers.median() #8.0
```

Correlation Matrix

```
#Step 1: Having 2 series or more
numbers1 = pd.Series([10, 8, 9, 1, 8, 0, 100])
numbers2 = pd.Series([7, 4, 0, 15, 6, 88, 120])

#Combining the 2 series
df = pd.concat([numbers1, numbers2], axis = 1)

#Correlation matrix function
df.corr()
```

Standard Deviation

```
numbers = pd.Series([10, 8, 9, 1, 8, 0, 100])
numbers.std() # 35.8
```

Shapiro-Wilk Test to test for Normal Test Distribution

```
numbers = pd.Series([10, 8, 9, 1, 8, 0, 100])
stat, p = st.shapiro(numbers.dropna())
print(p)
if p > 0.05:
  print('Sample looks Gaussian/Normal (fail to reject H0)')
else:
  print('Sample does not look Gaussian/Normal (reject H0)')
```

Standard Error of the Mean

```
numbers = pd.Series([10, 8, 9, 1, 8, 0, 100])
st.sem(numbers)
```

Confidence Interval of the Mean

```
numbers = pd.Series([10, 8, 9, 1, 8, 0, 100])
st.norm.interval(0.95, loc = numbers.mean(), scale = st.sem(numbers))
```

T-test

Step 1: Having 2 series

```
numbers1 = pd.Series([10, 8, 9, 1, 8, 0, 100])
numbers2 = pd.Series([7, 4, 0, 15, 6, 88, 120])
```

T-test calculation

```
stat, p = st.ttest_ind(numbers1, numbers2)
print(p)
if p > 0.05:
  print('Both countries have similar salaries (fail to reject H0)')
else:
  print('There is a difference in salaries (reject H0)')
```

Chi-Square Test

Step 1: Having 2 discrete series

```
numbers1 = pd.Series([0,1,0,1,1,1,0,0,1,0,0])
numbers2 = pd.Series([0,1,0,1,1,10,1,1,10,1])
```

First we create a cross tab

```
tab = pd.crosstab(index = numbers1,
            columns = numbers2)
```

Then execute the statistical test

```
chi2, p, dof, ex = st.chi2_contingency(tab, correction=False)
print(p)
if p > 0.05:
  print('There is no relationship (fail to reject H0)')
else:
  print('There is strong relationship (reject H0)')
```

Linear Regression

Step 1: Having 2 series, with ideally more than 30 elements each

```
numbers1 = pd.Series([10, 8, 9, 1, 8, 0, 100])
numbers2 = pd.Series([7, 4, 0, 15, 6, 88, 120])
```

Define dependent variable (y) and independent variable (X). The y must be continuous

```
y = numbers1
X = numbers2
```

Adding constant

```
X = sm.add_constant(X)
```

Regression model

```
model1 = sm.OLS(endog = y, exog = X).fit()
print(model1.summary())
```

Logistic Regression

Step 1: Having 2 series, with ideally more than 30 elements each

```
numbers1 = pd.Series([0,1,0,1,1,1,0,0,1,0,0])
numbers2 = pd.Series([7, 4, 0, 15, 6, 88, 120, 8, 9, 44, 23])
```

Define dependent variable (y) and independent variable (X). The y must be binary

```
y = numbers1
X = numbers2
```

Adding constant

```
X = sm.add_constant(X)
```

Training and test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2,
                                                    random_state = 1502)
```

Logistic Regression

```
model = sm.Logit(y_train, X_train).fit()
print(model.summary())
```

How to Read Logistic Regression Coefficients

```
def logistic_reader(coefficient):
  probability = round((np.exp(coefficient) - 1) * 100, 2)
  if probability > 0:
    print("The likelihood increases by", probability, "%")
  elif probability == 0:
    print("No impact")
  else:
    print("The likelihood decreases by", probability, "%")
```

Apply the function

```
logistic_reader(-0.5)
```

Assessing Logistic Regressions

Predictions

```
predictions = model.predict(X_test)
predictions = np.where(predictions > 0.5, 1, 0)
```

Classification report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

Multi-Linear Regression

Step 1: Having 3 series, with ideally more than 30 elements each

```
numbers1 = pd.Series([10, 8, 9, 1, 8, 0, 100])
numbers2 = pd.Series([7, 4, 0, 15, 6, 88, 120])
numbers3 = pd.Series([4, 6, 1, 8, 3, 6, 2])
```

Define dependent variable (y) and independent variable (X). The y must be continuous and you must have more than one independent variable

```
y = numbers1
X = pd.concat[(numbers2, numbers3], axis = 1)
```

Adding constant

```
X = sm.add_constant(X)
```

Training and test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2,
                                                    random_state = 1502)
```

Multi-linear regression

```
model = sm.OLS(y_train, X_train).fit()
print(model.summary())
```

Accuracy Assessment for Regression Problems

Predict

```
predictions = model.predict(X_test)
```

Accuracy assessment

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
print("MAE:", mean_absolute_error(y_test, predictions))
print("RMSE:", np.sqrt(mean_squared_error(y_test, predictions)))
```

Econometrics & Causal Inference

Definitions

Econometrics

The application of statistical methods to economic problems to derive relationships and outcomes.

Causal Inference

Study of causal relationships rather than just statistical ones.

Time Series Data

Data that happens consecutively through time in equally distanced time periods, i.e., every day, week, weekday, month, quarter, year.

Difference-in-differences

A causal inference approach that aims to mimic an experimental research design approach through the usage of at least one similar enough control group, thus minimizing omitted variable bias.

Experimental Research Design

Also called A/B test. It is a blind experiment where Groups A and B are randomly selected. One of the groups gets the treatment, and the other does not (or gets a placebo)

Omitted variable bias

When a model leaves out relevant variables (information).

Parallel Trends assumption

Assumption needed to use difference-in-differences. It means that both the treatment and control groups would have developed similarly, if not for the treatment.

Confounding Policy Change assumption

Assumption needed to use difference-in-differences. It means that there is one treatment (policy) applied and there is no other relevant change to any of the groups.

Google Causal Impact

A framework developed by Google applied to Difference-in-differences. It uses a space-state approach (information that varies in time and space) to improve causal inference. Additionally, it uses Bayesian inference for the causal simulations.

Matching

A quasi-experimental approach for when you don’t have a comparable control group (difference-in-differences is not possible). It finds observations/entities similar to the treatment group in the control group, leaving out the non-similar ones.

Unconfoundedess

Assumption needed to employ Matching. The variables used to perform matching have to be enough to fully describe the propensity to treatment by the individuals or entities.

Curse of dimensionality

Issues that arise when you have multiple variables, particularly binary, thus creating a problem high in dimensions. As a result, you can have dimensions that have low amount of observations and thus may not be representative of the dimension.

Common Support Region

Requirement for Matching. Matching can only be performed with similar observations. Thus, individuals in the treatment group that do not have commonalities with individuals of the control group must be left out.

Import Libraries

```
import pandas as pd
from causalimpact import CausalImpact
```

Data Retrieval

```
data = pd.read_csv('<https://bit.ly/causal_impact_cheat_sheet')[['y>', 'X']]
data.iloc[70:, 0] += 5
```

Pre and Post Period

```
pre_period = [0, 69]
post_period = [70, 99]
```

Causal Impact

```
ci = CausalImpact(data, pre_period, post_period)
print(ci.summary())
print(ci.summary(output='report'))
ci.plot()
```

Libraries

```
from causalinference import CausalModel
```

Isolate y, Treatment and Confounders

```
treat = df.treat.values
y = df.y
confounders = df.drop(columns = ["treat", "y"]).values
```

Matching Model

```
model = CausalModel(y, treat, confounders)
model.est_via_matching(bias_adj=True)
print(model.estimates)
```

Segmentation

Definitions

Segmentation

Process of splitting the target (market, customer) into groups of similar characteristics or actions.

Value-based segmentation

Process of segmenting customers into groups based on much value (revenue or profit) they bring to the company.

RFM

Segmentation approach where you segment customers based on their recency, frequency, and monetary KPIs. Recency is the time since the last purchase, Frequency is the number of orders in a given timeframe, and Monetary is the average basket the customer has.

Clustering

Unsupervised Learning concept. Process of dividing observation into groups of similar characteristics indicated by the model.

Unsupervised Learning

Machine Learning field for problems without a dependent variable. Opposite to Supervised Learning, where you have target/output/dependent variable.

Gaussian Mixture Model

Algorithm for Clustering which applies a probabilistic approach to the observations.

Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC)

Frameworks to pick the best model comprising of 2 criteria: simplicity and goodness of fit. The lower the value the better. The BIC penalizes more complex models.

RFM Python

Create Frequency Groups

```
df['F'] = pd.qcut(x = df['Frequency'], q = 4, labels = range(1, 5, 1))
df['M'] = pd.qcut(x = df['monetary'], q = 4, labels = range(1, 5, 1))
```

```
df['R'] = pd.qcut(x = df['Recency'], q = 4, labels = range(4, 0, -1))
df.head(5)
```

RFM Score

```
df['RFM'] = df[['R', 'F', 'M']].sum(axis = 1)
df.head(2)
```

Create the RFM function

```
def rfm_segment(df):
  if df['RFM'] >= 11:
    return 'SuperStar'
  elif ((df['RFM'] >=8) and (df['RFM'] < 11)):
    return 'Future Champion'
  elif ((df['RFM'] >=6) and (df['RFM'] < 8)):
    return 'High Potential'
  else:
    return 'Low Relevance'
```

Apply RFM function

```
df['RFM_level'] = df.apply(rfm_segment, axis = 1)
```

Looking into the segments

```
df.groupby('RFM_level').agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'monetary': ['mean', 'count']
})
```

Gaussian Mixture Model Python

Import libraries

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
```

Finding optimal number of clusters

```
n_components = np.arange(1,10)
```

Create GMM model

```
models = [GaussianMixture(n_components= n,
                          random_state = 1502).fit(df) for n in n_components]
```

Plot

```
plt.plot(n_components,
         [m.bic(df) for m in models],
         label = 'BIC')
plt.plot(n_components,
         [m.aic(df) for m in models],
         label = 'AIC')
plt.legend()
plt.xlabel('Number of Components')
```

Predictive Analytics

Definitions

Predictive Analytics

Processes of using data, statistics, machine learning, or any analytical framework to predict the future or outcomes.

Ensemble Learning

A framework where you use several Machine Learning or Analytical models for prediction. Minimizes the likelihood of a general bad outcome or high errors.

Random Forest

An ensemble Algorithm that combines multiple decision trees. It uses bagging and feature randomness for each Decision Tree. This means that each decision tree only uses a selection of the observation and the independent variables for its model.

Bagging

A process where you sample observations for the model. It provides stability, reduces noise and improves accuracy.

Decision Tree

A model that predicts based on decision rules (splits) inferred from the data. It is a map of possible outcomes.

Parameter Tuning

A process to find the best parameters for a given problem. It is the attempt of all combinations of parameters to be tuned to find the combination with the lowest error or maximum accuracy.

Structural Time Series

A framework where a time series is decomposed into at least 3 components: Trend, Seasonality, and External Regressors. Facebook Prophet also decomposes into Holiday effects.

Facebook Prophet

A Forecasting Model developed by Facebook, using a Structural Time Series approach.

Holiday_prior_scale

A parameter that dictates how much the Holidays affect the seasonality curve.

Seasonality_prior_scale

Strength of the seasonal cycles.

Changepoint_prior_scale

A parameter that states how easily the trend curve should change.

Additive seasonality

Type of seasonality in which the height of the fluctuation is the same no matter the trend.

Multiplicative seasonality

Type of seasonality in which the seasonal fluctuations are proportional to the trend.

Cross-validation

A process where the data is split into training and test set several times. Allows for model assessment in different circumstances. Usually combined with Parameter Tuning.

Random Forest in Python

Training and Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2,
                                                    random_state = 1502,
                                                    stratify = y)
```

Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
#from sklearn.ensemble import RandomForestRegressor
model = RandomForestClassifier(n_estimators = 300,
                               random_state = 1502)
model.fit(X_train, y_train)
```

Facebook Prophet in Python

Facebook Prophet Model

```
m = Prophet(growth= 'linear',
            yearly_seasonality = True,
            weekly_seasonality = True,
            daily_seasonality = False,
            holidays = holidays,
            seasonality_mode = "multiplicative",
            seasonality_prior_scale = 10,
            holidays_prior_scale = 10,
            changepoint_prior_scale = 0.05)
m.fit(df)
```

Create future dataframe

```
future = m.make_future_dataframe(periods = test_days,
                                 freq = 'D')
```

Forecasting

```
forecast = m.predict(future)
```

Visualization

```
m.plot(forecast);
m.plot_components(forecast);
```

Back To Top