**Python for Data Science Workshop by Manaranjan Pradhan:** <https://github.com/manaranjanp/Python_DS_2days>

**This repository contains the tutorials and exercises for the 2 days Python for Data Science Workshop.**

**The topics covered are**

* Understand Data Science Stack in Python
* Array and Matrix operations using Numpy library
* Data munging using Pandas Library
* Basic statistics Analysis & Hypothesis testing
* Creating charts and graphs using Visualization library
* Building Linear regression model
* Creating classification models to make predictions
* Clustering
* Text Analytics
* Using Sklearn library to run models and evaluate

**During the workshop, the following real world use cases are taught to the participants.**

* Analysing titanic survival data for insights
* Who is a better captain in cricket? Does the toss have any impact on the result of the game?
* How much to spend on Advertisement for a specific sales target? And target which media most: radio, newspaper or TV?
* Can we predict the selling price of a house?
* What are the chances a student’s application will be accepted by an university?
* Can we predict a customer churn, if we know his/her call patterns?
* Which customers would like what kind of beer?
* Can we find trends in tweets?
* Integrate with twitter APIs to stream live tweets
* Scrape data from website
* Can we understand the sentiments from customer reviews?

**Regression, classification, clustering.**

1. H1N1 Pandemic.

Statistical correlation of what people search?

Google flu trends: heat map -> could predict about 97% correct

* Data ===================> Generate Model

ML Algorithm

1. Recommendation system. -> Amazon etc.
2. Diaper-beer syndrome : Association rule Mining
3. Facebook relationship Breakups

**[ANALYTICS]: Descriptive, Predictive, and prescriptive.**

Descriptive: what and why from data…

Predictive: Regression and Classification (Supervised learning)

Prescriptive: How many cars to manufacture…??

**Python Stack for Data Science:**

R [ANALYSIS] and Python [Best Option☺]

**Python Stack for Data Science:**

Pandas -> prepare dataset

numpy (arrays and matrices)

scipy : Distribution, hypothesis test

sklearn: ML Algorithm

matplotlib, seaborn : statistical visualization

**Jupyter Notebook Usage**

How to start with Jupyter notebook??? Open anaconda GUI or open anaconda prompt and give below command

**Jupyter notebook --ip=\***

Types of cells:

Code cell:

Mark down cell for (documentation):

* ***Shift + enter for execute.***
* ***esc + a or esc+b or***
* ***esc +m -> markdown shell where u can write documentation, list and bold(\*), italic (\*\*)***
* (\*) Mark shows still executing

***pandas.read\_csv?*** -- see man page

**Dataset Source:**

Data sets: UCI Machine Learning Repository : <https://archive.ics.uci.edu/ml/index.php>

Kaggle Datasets : <https://www.kaggle.com/datasets> [Titanic]

**Pandas : Titanic\_Exploration\_1.0**

Read\_csv

E.g.

titanic\_df = pd.read\_csv('./data/titanic.csv')

titanic\_df.shape

pandas.core.frame.DataFrame

titanic\_df.head(5) -> view first 5 records

titanic\_df.info() -> total entries

titanic\_df.survived.value\_counts() -> distribution over particular feature

titanic\_df.survived.value\_counts(normalize = True) //in case if we have space in feature name use

titanic\_df**["survived"].**value\_counts()

pd.crosstab(titanic\_df.survived , titanic\_df.pclass, normalize='index')

pd.crosstab(titanic\_df.survived , titanic\_df.pclass, normalize=index)

pd.crosstab(titanic\_df.survived , titanic\_df.pclass, normalize='columns')

normalize = ‘index’ its rowwise percent

normalize = ‘column’ it’s columnwise percent

Values => Numerical [continuous/discrete] or categorical [nominal/ordinal]

**Matplotlib**

Histogram: (to see continuous value distribution)

plt.hist(titanic\_df.age, bins=20)

survived vs sex

pd.crosstab(titanic\_df.survived, titanic\_df.sex, margins = True)

Avg age of each pclass

titanic\_df.groupby('pclass')['age'].mean()

convert categorial to int

titanic\_df['gender'] = titanic\_df['sex'].map(lambda s: int(s == 'male'))

**Handling missing/invalid values**

verify data for missing/invalid values

cars\_df[-cars\_df.horsepower.str.isnumeric()]

cars\_df['horsepower'] = pd.to\_numeric(cars\_df.horsepower, errors='coarce')

**Signature:** pd**.**to\_numeric**(**arg**,** errors**='raise',** downcast**=None)**

**Docstring:**

Convert argument to a numeric type.

Parameters

----------

arg : list, tuple, 1-d array, or Series

errors : {'ignore', 'raise', 'coerce'}, default 'raise'

- If 'raise', then invalid parsing will raise an exception

- If 'coerce', then invalid parsing will be set as NaN

- If 'ignore', then invalid parsing will return the input

Overlapped plot

import matplotlib.pyplot as plt

import seaborn as sn

sn.distplot(cars\_df[cars\_df.origin == 1] ['mpg'], label = 'American')

sn.distplot(cars\_df[cars\_df.origin == 3] ['mpg'], label = 'Japanese')

**Different graph plots**

Histogram and distribution plot : one variable with continuous

Bar plot :One variable and categorical

Two variables :

Both Categorical : bar

One Cat and One continuous : Overlapped distribution/ histogram

Both continuous : scatter plot

Using scatter plot we can find correlation:

X inc y inc +ve correlation

X inc y dec –ve correlation

X inc y is random/constant

Pair plot to plot scatter plot for multiple functions in single graph

**Mathematical statistics**

Normal distributions: mean and variance matters

More variance, more fluctuations

Mean=avg=x(mean) = sum(xi)/i

Variance= 1/n[(xi-xmean)2]

Standard dev = sqroot(variance)

-1sd standard +1sdsd 68% of total -> 68%

-2sd left and +2sd on right -> 90% area of normal distribution curve

-3sd left and +3sd on right -> 99.7% area of normal distribution curve

<http://onlinestatbook.com/2/normal_distribution/areas_normal.html>

Normal/Gaussian Distribution

Cdf always starts with –infinity

**Predictive analysis : Regression**

Features should be quantifiable.

Historical evidence that factors influence outcome.

Multiple evidences.

Need sufficient observations

For dropping row/column

Axis = 0 is row

Axis = 1 is column

y-f(x1,x2,x3)

if y is numerical then its problem of regression else it’s a classification.

Training (80% )and test data(20%) :

Prepare model from training data and then test on testing data.

ML

Regression

Clustering

Classification

COvariance :

Pxy = ∑(Xi – Xmean) (Yi – Ymean)

--------------------------------

SDx . SDy

How to devide data :??

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Y | X1 | X2 | X3 | X4 | X5 | X6 |
| **Y\_train** | **X\_train** | | | | | |
| **Y\_test** | **X\_test** | | | | | |

**80 percent of total data**

While deriving equation for linear regression function, ML algo uses cost function which considers minimum error for points from regression line.

MSE (Mean squared Error) :

MSE = 1/n (∑Yi - Yp)2 ….Minimize the MSE\*\*\*\*

Code Snippet :

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

x\_train, x\_test,\

y\_train, y\_test = train\_test\_split(sales\_df[X\_features], sales\_df.Sales, train\_size = 0.8, random\_state = 100)

Build Linear Regression model:

from sklearn.linear\_model import LinearRegression

lreg\_v1 = LinearRegression()

lreg\_v1.fit(x\_train,y\_train)

**lreg\_v1.intercept\_**

**lreg\_v1.coef\_**

**y =ax + b**

**MSE or RMSE ?? people choose RMSE**

***R2 score is delta when Error when model built – Error when model not built 0 if model is bad and > 0 if good***

Feature Engineering :

Cross validation strategy : devide training data into different number of folds

Cross validation :

score\_v1 = cross\_val\_score(LinearRegression(),x\_train,y\_train,cv=5,scoring='r2')

score\_v2.mean()

score\_v2.std()

**Save model and load using numpy**

**Classification**

Sigmoid function :

Probability of output is function of inputs.

P(y=1) = f(X1,X2…….)

KNN : K nearest Neighbours

(We can use Euclidean distance for finding KNN)

**Encoding categorical variables**

hr\_encoded\_df = pd.get\_dummies(hr\_df, columns=['sales','salary'])

knn\_v1 = KNeighborsClassifier(n\_neighbors=5) >> hyperparameter which is parameter other than data

**Installation for Decision tree**

* Install Graphviz (graphviz 2.38 msi)
* Shut down all notebooks
* Stop jupyter notebook
* Add graphviz to bin folder to PATH
* Restart Anaconda promt
* Install pydotplus
* Start jupyter notebook

conda install -c conda-forge pydotplus

conda install -c anaconda graphviz

Steps for a data model

1.       Identify the outcome variable and input factors

2.       If any of the factors is categorical, create dummy variables

3.       Split data set into training and test

4.       Build a model using training data set, using cross validation approach

5.       Check accuracy of the model (R^2)

6.       If R^2 is less than expected

1.       Feature engineering

2.       Different model

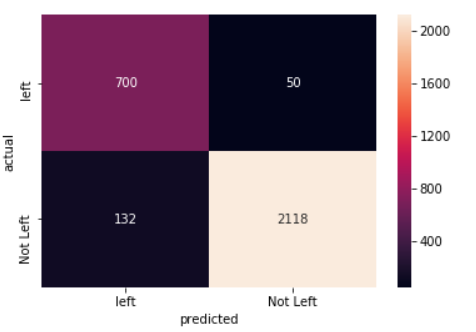
3.       Add more factors

7.       Repeat step 4-6

8.       Select model and test against test data set

9.       Calculate RMSE and R^2

Confusion Matrix:



|  |  |
| --- | --- |
| TP | FN |
| FP | TN |

**TP+TN/TP+TN+FP+FN**

***FP and FN cannot be reduced simultaneously.***

***Precision : TP/TP+FP***

***Recall : TP/TP+FN***

***F1score : 2PR/P+R***

# the  below diagram represents  as:

#############

#  TP # FN #

############

#  FP # TN #

###########

# where T True,  F false  N -ve and  P +ve

#  Precision will be  ( TP / ( TP+FP ) )

#  Recall  will be  ( TP / ( TP + FN ) )

# H-score  will be ( 2PR / ( P+ R) )

# If the  business  Precision is important than recall  they will look for  less  FP and vise versa.

# If both are  important business will use  H-score

**Decision Tree Algorithms**

***GINI Index:*** *1-∑(Pi2)*

*Information Gain: 1-∑(PiLogkPi)2*

**Clustering (K-means)**

Unsupervised Learning

**Plots**

***Categorical and numeric : distplot***

***Both Numeric: Scatter plot***

***Pair plot : multiple numeric columns***

***Heat Map: Correlation***

***Cluster Map: clustering***

**Some of the code snippets:**

**Comments and documentation:**

***# HEADING 1***

***## 2222222***

***### 33333333***

***list***

***- item 1***

***- item 2***

***This is \*Italic\* and this is \*\*Bold\*\****

**Read CSV :**

cars\_df = pd.read\_csv('./data/Auto.csv')

**Top Records:**

cars\_df.head()

**Unique records in a column:**

cars\_df.year.unique()

**Dataset Information:**

cars\_df.info()

**Convert Errors to numeric value:**

cars\_df['horsepower'] = pd.to\_numeric(cars\_df.horsepower, errors='coarce')

**Dist plot:**

import matplotlib.pyplot as plt

import seaborn as sn

sn.distplot(cars\_df[cars\_df.origin == 1] ['mpg'], label = 'American')

sn.distplot(cars\_df[cars\_df.origin == 3] ['mpg'], label = 'Japanese')

#to print labels

plt.legend()

sn.distplot(hr\_df[hr\_df.left == 1] ['time\_spend\_company'], label = 'left')

sn.distplot(hr\_df[hr\_df.left == 0] ['time\_spend\_company'], label = 'stayed')

plt.legend()

**Scatter Plot:**

plt.scatter(cars\_df.horsepower, cars\_df.mpg)

plt.xlabel('horsepower')

plt.ylabel('mpg')

Pair Plot:

cols =['mpg','horsepower','acceleration','displacement']

cars\_df\_v1 = cars\_df[cols]

sn.pairplot(cars\_df\_v1)

**Correlation between features :**

cols =['mpg','horsepower','acceleration','displacement']

cars\_df\_v1 = cars\_df[cols]

cars\_df\_v1.corr()

**Heat Map:**

sn.heatmap(cars\_df\_v1.corr(), annot=True)

sn.heatmap(cars\_df\_v1.corr(),

annot=True,

cmap=sn.diverging\_palette(10, 240, n=10))

**Scaling up the features:**

from sklearn.preprocessing import StandardScaler

scalar = StandardScaler()

x\_scaled = scalar.fit\_transform(beer\_df[['calories','sodium','alcohol','cost']])

**Cluster Map:**

import matplotlib.pyplot as plt

import seaborn as sn

sn.clustermap(x\_scaled) >>>> from this find number of clusters for k means

**Catplot:**

sn.catplot(data=hr\_df,

x='time\_spend\_company',

kind='count',

hue='left')

sn.catplot(data=hr\_df,

x='salary',

kind='count',

hue='left')

**CROSSTAB:**

pd.crosstab(titanic\_df.survived, titanic\_df.sex, margins = True)

pd.crosstab(titanic\_df.survived , titanic\_df.pclass, margins = True)

pd.crosstab(titanic\_df.survived , titanic\_df.pclass, normalize='index')

pd.crosstab(titanic\_df.survived , titanic\_df.pclass, normalize='columns')

**GroupBy:**

titanic\_df.groupby('pclass')['age'].mean()

**Conversion:**

titanic\_df['gender'] = titanic\_df['sex'].map(lambda s: int(s == 'male'))

**\*\*\* conditions**

children\_df= titanic\_df[titanic\_df.age < 10]

**K-means:**

from sklearn.cluster import KMeans

kmeans = KMeans(6)

kmeans.fit(x\_scaled, ran)

beer\_df['cluster'] = kmeans.labels\_

beer\_df[beer\_df.cluster == 0]

**plot points:**

import pandas as pd

import numpy as np

import sklearn

import matplotlib as plt

import seaborn as sn

%matplotlib inline

y = np.array( [3, 6, 2, 5, 6, 2, 8, 9, 12, 23 ] )

x = np.array( [1, 2, 4, 8, 7, 1, 8, 23, 16, 11] )

df = pd.DataFrame( { "x": x, "y": y } )

sn.lmplot( y = "y", x = "x", data = df )

**Convert binary categorical variable to numericL:**

heart\_df['family'] = heart\_df['famhist'].map(lambda s: int(s == 'Present'))

**get feature list:**

x\_features = list(heart\_df.columns)

**Remove features from list:**

x\_features.remove('row.names')

**Split data set:**

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

x\_train, x\_test,\

y\_train, y\_test = train\_test\_split(heart\_df[x\_features],

heart\_df.chd,

train\_size=0.8,

random\_state = 100)

**Linear Regression:**

from sklearn.linear\_model import LinearRegression

lreg\_v1 = LinearRegression()

lreg\_v1.fit(x\_train,y\_train)

dict(zip(X\_features, lreg\_v1.coef\_))

y\_pred = lreg\_v1.predict(x\_test)

y\_df\_v1 = pd.DataFrame({'actual' : y\_test,

'predicted' : y\_pred,

'residual' : y\_test - y\_pred})

**MSE :**

from sklearn.metrics import mean\_squared\_error

mse\_v1 = mean\_squared\_error(y\_df\_v1.actual,

y\_df\_v1.predicted)

**RMSE:**

import numpy as np

rmse\_v1 = np.sqrt(mse\_v1)

**R2 Score:**

from sklearn.metrics import r2\_score

r2\_score(y\_df\_v1.actual, y\_df\_v1.predicted)

Creating new variable as a function of existing two variables:

x\_train['tv\_radio'] = x\_train['TV'] \* x\_train['Radio']

x\_test['tv\_radio'] = x\_test['TV'] \* x\_test['Radio']

X\_features = ['TV','Radio','Newspaper','tv\_radio']

**Cross Validation:**

from sklearn.model\_selection import cross\_val\_score

score\_v2 = cross\_val\_score(LinearRegression(),x\_train,y\_train,cv=5,scoring='r2')

score\_v2

score\_v2.mean()

score\_v2.std()

**Building KNN Model:**

from sklearn.neighbors import KNeighborsClassifier

knn\_v1 = KNeighborsClassifier(n\_neighbors=5)

knn\_v1.fit(x\_train,y\_train)

Predict and evaluate KNN Model:

y\_pred\_knn\_v1 = knn\_v1.predict(x\_test)

y\_knn\_v1 = pd.DataFrame({'actual': y\_test,

'predicted' : y\_pred\_knn\_v1})

Confusion Matric for predicted vs actual:

from sklearn.metrics import confusion\_matrix

cm\_v1 = confusion\_matrix(y\_knn\_v1.actual,y\_knn\_v1.predicted, [1,0])

sn.heatmap(cm\_v1, fmt="0.0f", xticklabels=['left', 'Not Left'], yticklabels=['left', 'Not Left'], annot=True)

plt.xlabel('predicted')

plt.ylabel('actual')

**custom probabililty:**

knn\_v1.predict\_proba(x\_test)[0:10]

y\_knn\_v1['prob'] = knn\_v1.predict\_proba(x\_test)[:,1]

y\_knn\_v1['new\_Predict'] = y\_knn\_v1.prob.map(lambda x: 1 if x >= 0.2 else 0)

cm\_v1\_new = confusion\_matrix(y\_knn\_v1.actual,y\_knn\_v1.new\_Predict, [1,0])

sn.heatmap(cm\_v1\_new, fmt="0.0f", xticklabels=['left', 'Not Left'], yticklabels=['left', 'Not Left'], annot=True)

plt.xlabel('new predicted')

plt.ylabel('actual')

**Print classification Report:**

from sklearn.metrics import classification\_report

print(classification\_report(y\_knn\_v1.actual, y\_knn\_v1.final\_pred))

**Finding model using grid Search: (decision tree)**

tree\_v2 = DecisionTreeClassifier()

from sklearn.model\_selection import GridSearchCV

tree\_params = {'max\_depth' : range(3,11),

'criterion' : ['entropy','gini']}

tree\_grid = GridSearchCV(tree\_v2, param\_grid=tree\_params, cv=10, scoring='recall')

tree\_grid.fit(x\_train, y\_train)

tree\_grid.best\_score\_

tree\_grid.best\_params\_

tree\_grid.best\_estimator\_

**Finding model using grid Search: (KNN)**

knn\_v2 = KNeighborsClassifier()

knn\_params = {'n\_neighbors' : range(5,15),

'metric' : ['euclidean','minkowski']}

knn\_grid = GridSearchCV(knn\_v2, param\_grid=knn\_params, cv=10, scoring='recall')

knn\_grid.fit(x\_train, y\_train)

knn\_grid.best\_estimator\_

**Decision Tree classifier:**

from sklearn.tree import DecisionTreeClassifier

tree\_v1 = DecisionTreeClassifier(max\_depth=5)

tree\_v1.fit(x\_train, y\_train)

y\_pred\_tree\_v1 = tree\_v1.predict(x\_test)

**Confusion Matrix:**

from sklearn.metrics import confusion\_matrix

import seaborn as sn

cm\_tree\_v1 = confusion\_matrix(y\_test, y\_pred\_tree\_v1, [1,0])

sn.heatmap(cm\_tree\_v1, fmt="0.0f", xticklabels=['problem', 'Not problem'], yticklabels=['problem', 'Not problem'], annot=True)

plt.xlabel('new predicted')

plt.ylabel('actual')

**Export decision tree:**

from sklearn.tree import export\_graphviz

tree = export\_graphviz(tree\_v1, out\_file='tree\_heart.odt',feature\_names=x\_features, class\_names=['Not Left','Left'], filled=True)

import pydotplus as pdt

graph = pdt.graphviz.graph\_from\_dot\_file('tree\_heart.odt')

graph.write\_jpg('tree\_heart.jpg')

**Read image:**

from IPython.display import Image

Image(filename='tree\_heart.jpg')

**Finding Optimal Cutoff:**

***FP\_Cost = 1.0***

***FM\_Cost = 5.0***

y\_tree\_v1 = pd.DataFrame({'actual': y\_test,

'predicted' : y\_pred\_tree\_v1})

y\_tree\_v1['prob'] = tree\_v1.predict\_proba(x\_test)[:,1]

def get\_Cost(cutoff) :

y\_tree\_v1['new\_Predict'] = y\_tree\_v1.prob.map(lambda x: 1 if x >= cutoff else 0)

cm\_v1\_new = confusion\_matrix(y\_tree\_v1.actual,y\_tree\_v1.new\_Predict, [1,0])

return FP\_Cost \* cm\_v1\_new[1][0] + FM\_Cost \* cm\_v1\_new[0][1]

import numpy as np

cutoff\_list = np.arange(0.1,0.8,0.01)

cost\_list = [get\_Cost(x) for x in cutoff\_list]

cost\_df = pd.DataFrame({'cutoff' : cutoff\_list,

'cost' : cost\_list})

cost\_df.sort\_values('cost', ascending=True)[0:10]

y\_tree\_v1['final\_pred'] = y\_tree\_v1.prob.map(lambda x: 1 if x >= 0.10 else 0)

**Encoding categorical Variables:**

hr\_encoded\_df = pd.get\_dummies(hr\_df, columns=['sales','salary'])

x\_features = list(hr\_encoded\_df.columns)

**Stock Gain:**

tcs\_df['gain'] = (tcs\_df['Close Price'] - tcs\_df['Open Price']) \* 100 / tcs\_df['Open Price']

import matplotlib.pyplot as plt

import seaborn as sn

plt.plot(tcs\_df.gain)

sn.distplot(tcs\_df.gain, label = 'TCS')

sn.distplot(dlf\_df.gain, label = 'DLF')

plt.legend()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* statistics \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

from scipy import stats

stats.norm.interval(0.65, tcs\_df.gain.mean(),tcs\_df.gain.std())

stats.norm.cdf(-3,tcs\_df.gain.mean(),tcs\_df.gain.std())

stats.norm.cdf(-3,dlf\_df.gain.mean(),dlf\_df.gain.std())

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Hostogram:**

titanic\_df.survived.value\_counts(normalize = True) \* 100 🡪 percent

titanic\_df["survived"].value\_counts() 🡪 numbers

import matplotlib.pyplot as plt

plt.hist(titanic\_df.age, bins=20)

plt.title('Age Distribution')

plt.xlabel('Age')

plt.ylabel('Frequency')

plt.savefig('age.png')

**Saving Model:**

from sklearn.externals import joblib

joblib.dump(lreg\_v2, 'sales.pkl')

!ls –al >> show dir

**Load model and predict:**

from sklearn.externals import joblib

sales\_model=joblib.load('sales.pkl')

sales\_model.intercept\_

sales\_model.coef\_

sales\_model.predict([[50,20,10,1000]])

**Jupyter notebooks shortcuts**

The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code or text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin.

#### **Command Mode (press Esc to enable)Edit Shortcuts**

F: find and replace

Ctrl-Shift-F: open the command palette

Ctrl-Shift-P: open the command palette

Enter: enter edit mode

P: open the command palette

Shift-Enter: run cell, select below

Ctrl-Enter: run selected cells

Alt-Enter: run cell and insert below

Y: change cell to code

M: change cell to markdown

R: change cell to raw

1: change cell to heading 1

2: change cell to heading 2

3: change cell to heading 3

4: change cell to heading 4

5: change cell to heading 5

6: change cell to heading 6

K: select cell above

Up: select cell above

Down: select cell below

J: select cell below

Shift-K: extend selected cells above

Shift-Up: extend selected cells above

Shift-Down: extend selected cells below

Shift-J: extend selected cells below

A: insert cell above

B: insert cell below

X: cut selected cells

C: copy selected cells

Shift-V: paste cells above

V: paste cells below

Z: undo cell deletion

D,D: delete selected cells

Shift-M: merge selected cells, or current cell with cell below if only one cell is selected

Ctrl-S: Save and Checkpoint

S: Save and Checkpoint

L: toggle line numbers

O: toggle output of selected cells

Shift-O: toggle output scrolling of selected cells

H: show keyboard shortcuts

I,I: interrupt the kernel

0,0: restart the kernel (with dialog)

Esc: close the pager

Q: close the pager

Shift-L: toggles line numbers in all cells, and persist the setting

Shift-Space: scroll notebook up

Space: scroll notebook down

#### **Edit Mode (press Enter to enable)**

Tab: code completion or indent

Shift-Tab: tooltip

Ctrl-]: indent

Ctrl-[: dedent

Ctrl-A: select all

Ctrl-Z: undo

Ctrl-/: comment

Ctrl-D: delete whole line

Ctrl-U: undo selection

Insert: toggle overwrite flag

Ctrl-Home: go to cell start

Ctrl-Up: go to cell start

Ctrl-End: go to cell end

Ctrl-Down: go to cell end

Ctrl-Left: go one word left

Ctrl-Right: go one word right

Ctrl-Backspace: delete word before

Ctrl-Delete: delete word after

Ctrl-Y: redo

Alt-U: redo selection

Ctrl-M: enter command mode

Ctrl-Shift-F: open the command palette

Ctrl-Shift-P: open the command palette

Esc: enter command mode

Shift-Enter: run cell, select below

Ctrl-Enter: run selected cells

Alt-Enter: run cell and insert below

Ctrl-Shift-Minus: split cell at cursor

Ctrl-S: Save and Checkpoint

Down: move cursor down

Up: move cursor up