# PYTHON DATA SCIENCE CHEAT SHEET

## IMPORTING DATA

```
df = pd.read_csv(filename[,sep=','])
df = pd.read_excel(filename)
df = pd.read_sql(sqlcommand,connection)
```

## EXPORTING DATA

```
df.to_csv(filename,sep=',')
```

## VIEWING/EXPLORING DATA

`print(df)`
`df.head(n)` – show first n lines
`df.tail(n)` – show last n lines
`df.shape()` – show nr of rows and cols
`df.describe()` – summary statistics for numerical columns
`df[df['col']].isnull()` – find empty lines in column

## SELECTION

`df['col']` – returns column col as Pandas Series
`df[['col1','col2','col3']]` – returns columns col1,col2,col3 as Pandas DataFrame
`df.index` – returns index of DataFrame as Pandas Series

## DATA CLEANING

`df['col'] = df['col'].str.strip()` – strip off blanks
`df['col'] = df['col'].str.lower()` – to lowercase
`df['col'] = df['col'].map(myfunction)` – apply myfunction to all elements in column
`df = df.dropna()` – drop all lines that contain null values
`df = df.drop('col',axis=1)` – drop one column
`df = df.drop_duplicates(subset = ['col1','col2'],keep='last')` – drop rows that contain duplicate values for the combination 'col1','col2'. `keep` can be 'last' or 'first'.

## DATA MANIPULATION

`np.sqrt(a) or np.sqrt(df['col'])` – returns square root of value or column
`df['col'] = np.where(condition,value if true,value if false)` – immediate if
`df = pd.get_dummies(df,columns=['col'], prefix=['col'])` – one hot encoding

## FILTER, SORT & GROUPBY

`df[df['col'] > 5]` – return rows where value of col is greater than 5
`df[(df['col1'] > 5) & (df['col2'] < 7)]`
`df.groupby('col')` – group by one column
`df.groupby(['col1','col2'])` – group by multiple columns
`df.groupby('col1')['col2'].mean()` – mean of the values in col2 grouped by the values in col1. min() can be replaced by max,count,mean,sum,…
`df.groupby(['col1,'col2'])[col3'].min().unstack().fillna(0)` – pivot table with col1 in rows, col2 in columns and minimum of col3 in cells.
`df.sort_values(by='col',ascending=False)` – sort dataframe by values in col.

## JOIN/COMBINE

`df1.append(df2)` – add the rows in df2 to the end of df1 (cols should be identical)
`pd.concat([df1,df2],axis=1)` – add the columns of df2 to the right of df1 (number of rows should be identical)
`df1 = pd.merge(df1,df2,left_on='col1',right_on='col2',how='inner')` – SQL style join of the columns in df1 and df2 where the rows for col have identical values. `how` can be 'inner', 'left', 'right' or 'outer' (= SQL full outer join).

## MACHINE LEARNING

```
# split in 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=0.30)

# classification: imports
from sklearn.naive_bayes import GaussianNB,MultinomialNB, RandomForestClassifier
# classification: construct model
model = GaussianNB() – Gaussian Naïve Bayes
model = MutilnomialNB()
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train,y_train) –
```
fit data to model

```
# word count
from sklearn.feature_extraction.text import TfidfVectorizer
vec = TfidfVectorizer()

# pipeline of classifiers
model = make_pipeline(Classifier1, Classifier2, …)

pd.DataFrame(model.feature_importances_,columns=['Importance'],index=X_train.columns).sort_values(by='Importance',ascending=False) –
```
feature importances for decision trees

## ACCURACY METRICS

`from sklearn.metrics import accuracy_score, mean_squared_error,mean_absolute_error,r2_score`
`accuracy_score(y1, y2)` – categorical data
`mean_squared_error(y1,y2)` – continuous data