Data Science Cheat Sheet

Imports

```
import pandas as pd
import numpy as np
import altair as alt
import seaborn as sns

# for JSON
import urllib3
import json

# for SQL
import sys
import datadotworld as dw

# for machine learning
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

Load in a JSON file

```
http = urllib3.PoolManager()
response = http.request('GET', url)
DataFrame_json = json.loads(response.data.decode('utf-8'))
DataFrame = pd.json_normalize(DataFrame_json)
```

Pandas Basic Functions

```
pd.read_csv(url) # turn a csv file from a link into a dataframe/tibble
DataFrame.query('column_name == value & column_name == value') # filter by
rows
DataFrame.filter([column_name, column_name, column_name]) # filter
by/select columns
DataFrame.dtypes # shows you the variable types for each column
DataFrame.info() # provides a print out of data types on other useful
information about your pandas data frame
DataFrame.assign(
    new_column_name = lambda x: x.column1 // 100,
    new_column_name = lambda x: x.column2 - x.column1
) # create new columns, generally use lambda to edit current columns
DataFrame.rename(columns = {'year': 'YEAR', 'month':'MONTH'}) # rename a
column
new = DataFrame.groupby(['column1', 'column2'])
new.agg(delay = ('column3', np.mean)).reset_index() # groupby
DataFrame.drop(columns='column_name')
DataFrame.drop(['column1', 'column2', 'column3'], axis = 1)

flights_2 = flights.groupby(['airport_code']).sum().reset_index() #simpler
groupby without agg
DataFrame['month'].replace('n/a', np.NaN) # replace all specific values in
a column with another value
DataFrame['column'].fillna(method='ffill') # fill all np.NaN values with
whatever is behind it
```

Use Dataworld and SQL

```
SQL_query = '''
SELECT teams.franchid,
    TeamsFranchises.franchname,
    SUM (teams.w) as wins,
    SUM (Salaries.salary) as all_dollars_payed,
    SUM (teams.attendance) as all_time_attendance,
    SUM (Salaries.salary) / SUM (teams.attendance) as cost_of_attendance,
    SUM (teams.attendance) / (SUM (teams.w) * 10000) as loyalty
FROM teams
    JOIN Salaries ON Salaries.teamid = teams.teamid
    JOIN TeamsFranchises ON teams.franchid = TeamsFranchises.franchid
WHERE Teams.yearid > 1999
GROUP BY Teams.franchid
ORDER BY cost_of_attendance DESC
'''
DataFrame = dw.query('dataworld_file_path, SQL_query)
```

Altair Chart

```
(alt.Chart(DataFrame)
    .encode(
        alt.X( # edit the x-axis
            'column1', # select column for x-axis
            axis=alt.Axis(
                format='.4', # 4 digits no comma
                title="Insert Column Name"), # change column title
            scale = alt.Scale(domain = (1980, 2015)),
            sort = ['column2, column1] # change column order
        ),
        alt.Y( # edit the y-axis
            'column2', # select column for y-axis
        scale = alt.Scale(domain = (0, 220))),
        color = 'column3' # create a Z-axis represented by different
colors
    ).properties(
    title='Insert Chart Title' # change chart title
    )
    .mark_line() # type of chart
```

Seaborn

```
correlation = sns.pairplot(DataFrame, hue = 'column')
```

Machine Learning

```
X_train, X_test, y_train, y_test = train_test_split(
    X_pred,
    y_pred,
    test_size = .34,
    random_state = 76)

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

score = accuracy_score(y_test,y_pred)

df_features = pd.DataFrame(
    {'f_names': X_train.columns,
    'f_values': clf.feature_importances_}).sort_values('f_values',
ascending = False)
```