Nanshan Li

UNI: nl2643

Date: Feb 3 2019

Applied Machine Learning Homework 1

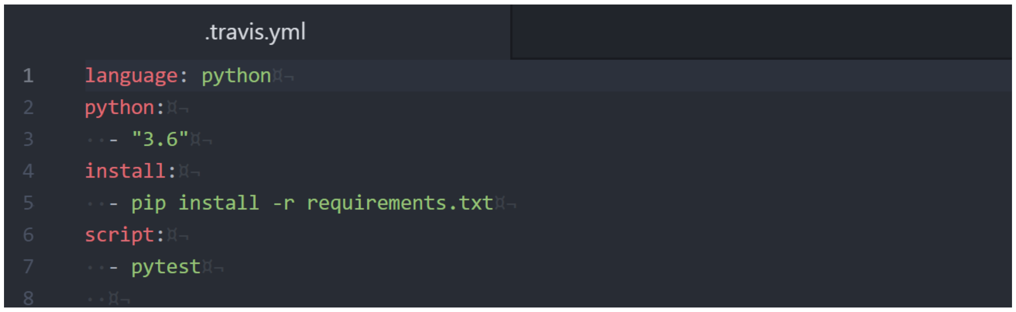
Task 0

I Nanshan Li, UNI nl2643, consent to audio recording made during the lecture to be posted publicly under a CC-0 license.

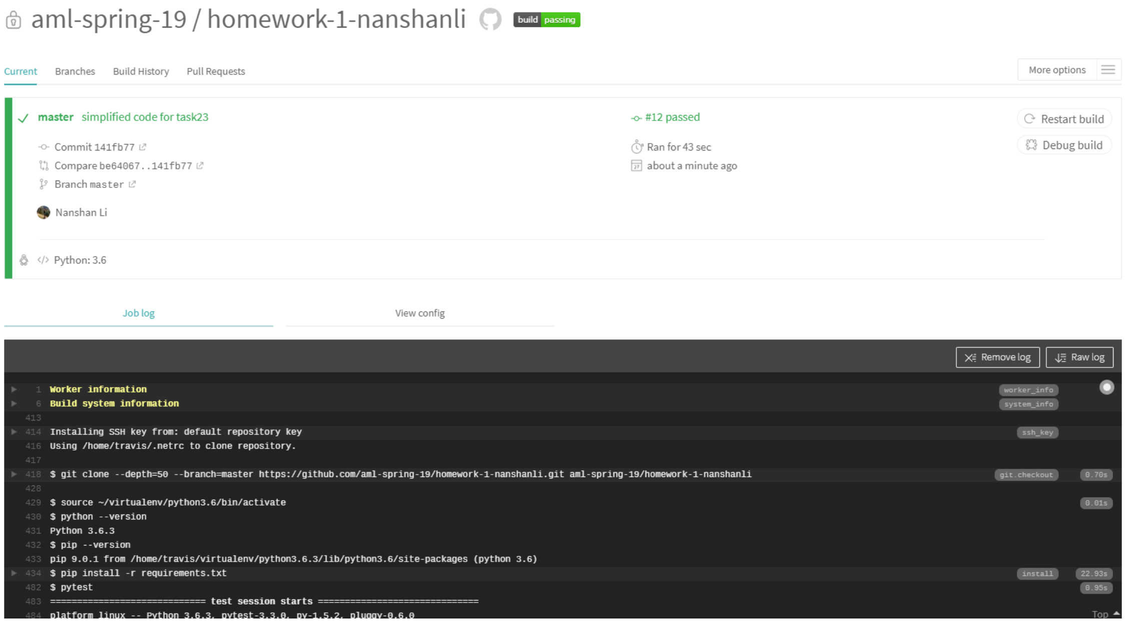
This includes any accidental recordings of my voice.

Nanshan Li

Task 1.1


.travis.yml

```yaml
language: python
python:
  - "3.6"
install:
  - pip install -r requirements.txt
script:
  - pytest
  
```


aml-spring-19 / homework-1-nanshanli

master simplified code for task23

Commit 141fb77
Compare be64067..141fb77
Branch master

Nanshan Li

#12 passed
Ran for 43 sec
about a minute ago

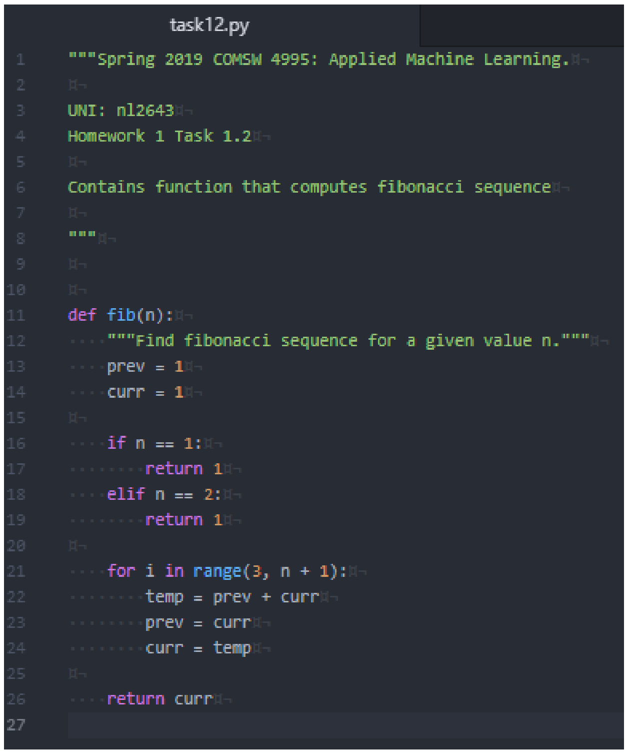
Python: 3.6

```
  1 Worker information                                                                  worker_info
  6 Build system information                                                            system_info
413
414 Installing SSH key from: default repository key                                     ssh_key
416 Using /home/travis/.netrc to clone repository.
417
418 $ git clone --depth=50 --branch=master https://github.com/aml-spring-19/homework-1-nanshanli.git aml-spring-19/homework-1-nanshanli    git.checkout  0.70s
428
429 $ source ~/virtualenv/python3.6/bin/activate                                        0.01s
430 $ python --version
431 Python 3.6.3
432 $ pip --version
433 pip 9.0.1 from /home/travis/virtualenv/python3.6.3/lib/python3.6/site-packages (python 3.6)
434 $ pip install -r requirements.txt                                                   install  22.93s
482 $ pytest                                                                            0.95s
483 ============================= test session starts ==============================
484 platform linux -- Python 3.6.3, pytest-3.3.0, py-1.5.2, pluggy-0.6.0
```

https://travis-ci.com/aml-spring-19/homework-1-nanshanli

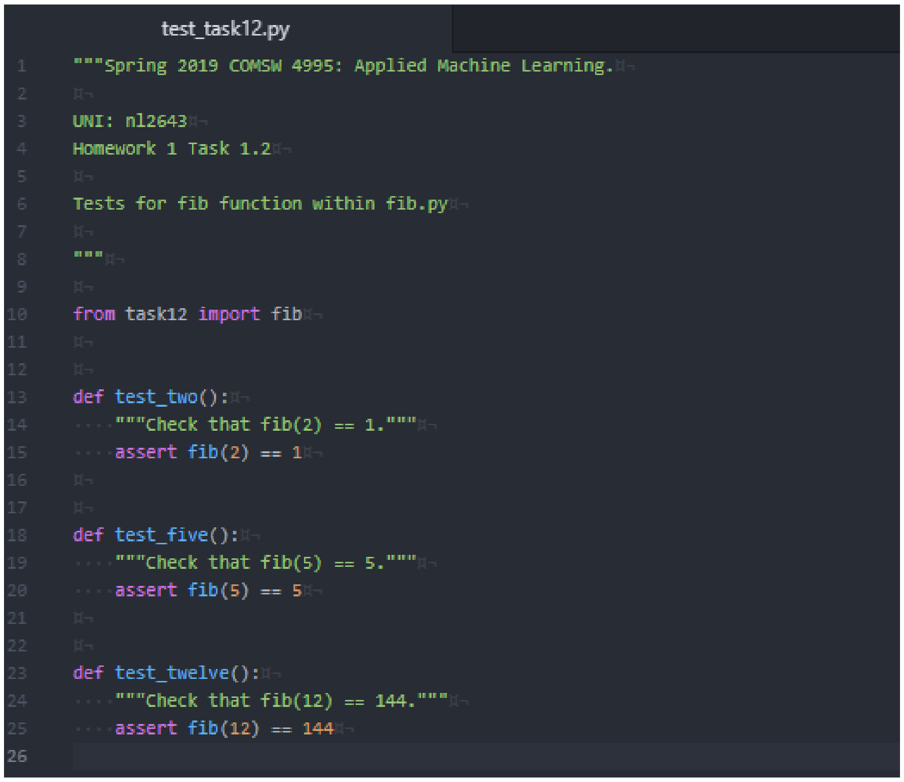
Task 1.2

task12.py



```python
"""Spring 2019 COMSW 4995: Applied Machine Learning.

UNI: nl2643
Homework 1 Task 1.2

Contains function that computes fibonacci sequence

"""


def fib(n):
    """Find fibonacci sequence for a given value n."""
    prev = 1
    curr = 1

    if n == 1:
        return 1
    elif n == 2:
        return 1

    for i in range(3, n + 1):
        temp = prev + curr
        prev = curr
        curr = temp

    return curr

```

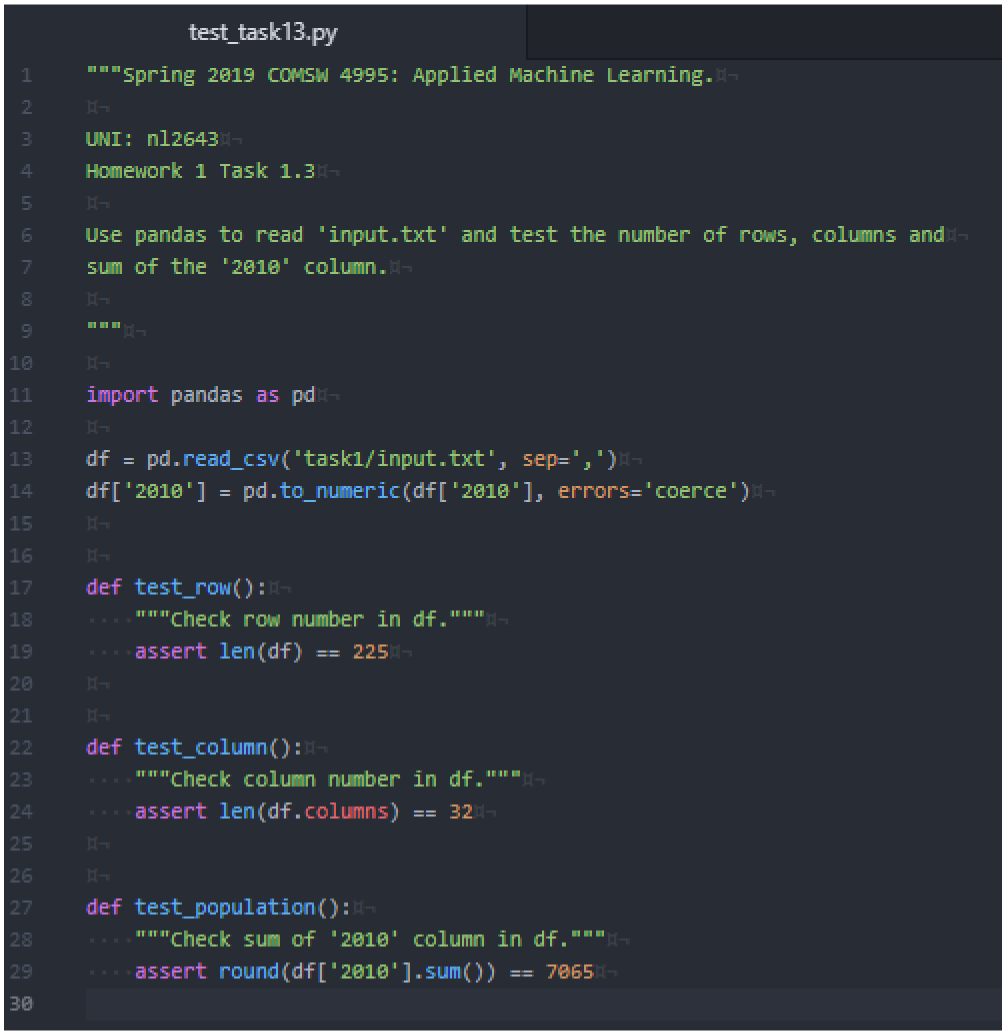
test_task12.py



```python
"""Spring 2019 COMSW 4995: Applied Machine Learning.

UNI: nl2643
Homework 1 Task 1.2

Tests for fib function within fib.py

"""

from task12 import fib


def test_two():
    """Check that fib(2) == 1."""
    assert fib(2) == 1


def test_five():
    """Check that fib(5) == 5."""
    assert fib(5) == 5


def test_twelve():
    """Check that fib(12) == 144."""
    assert fib(12) == 144

```

Task 1.3

test_task13.py

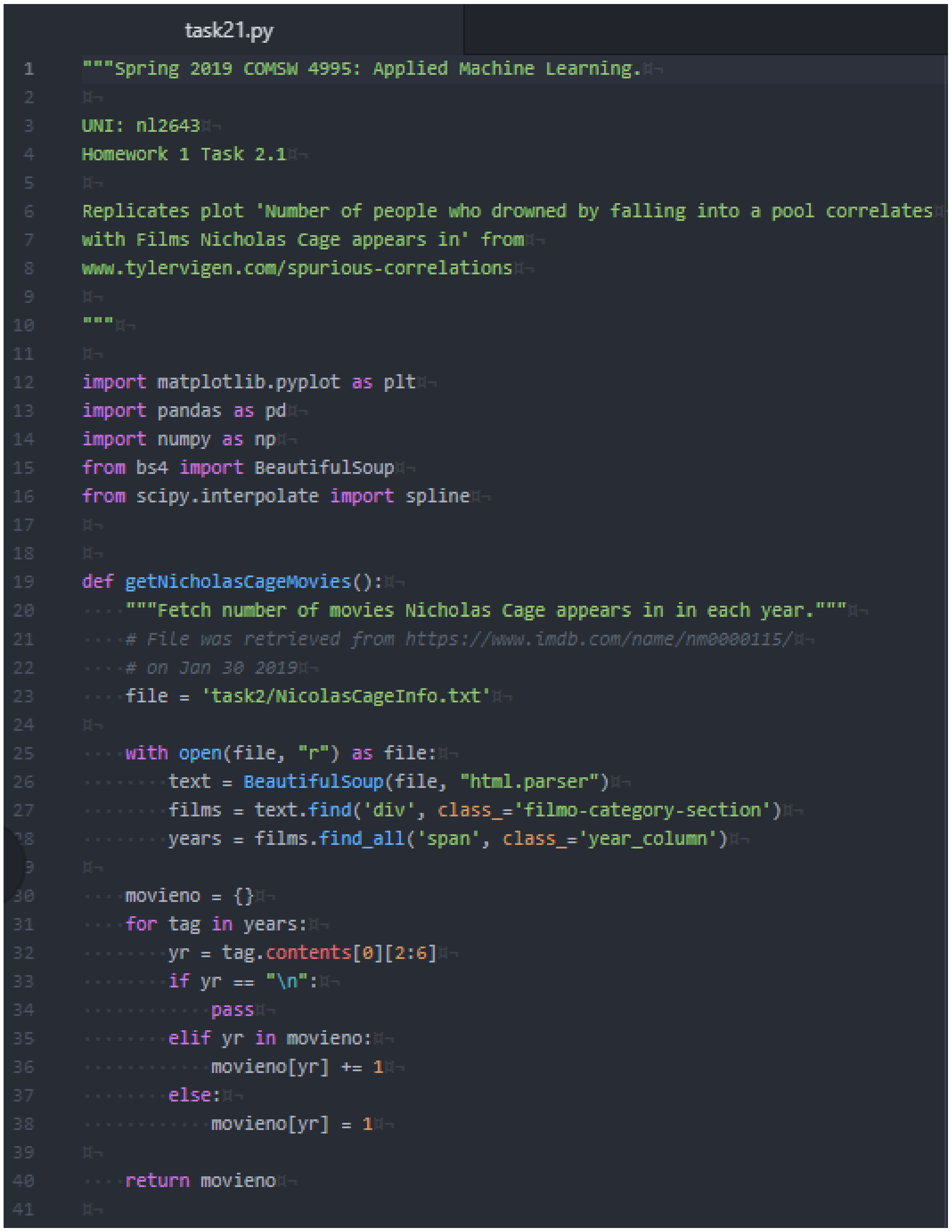


test_task13.py

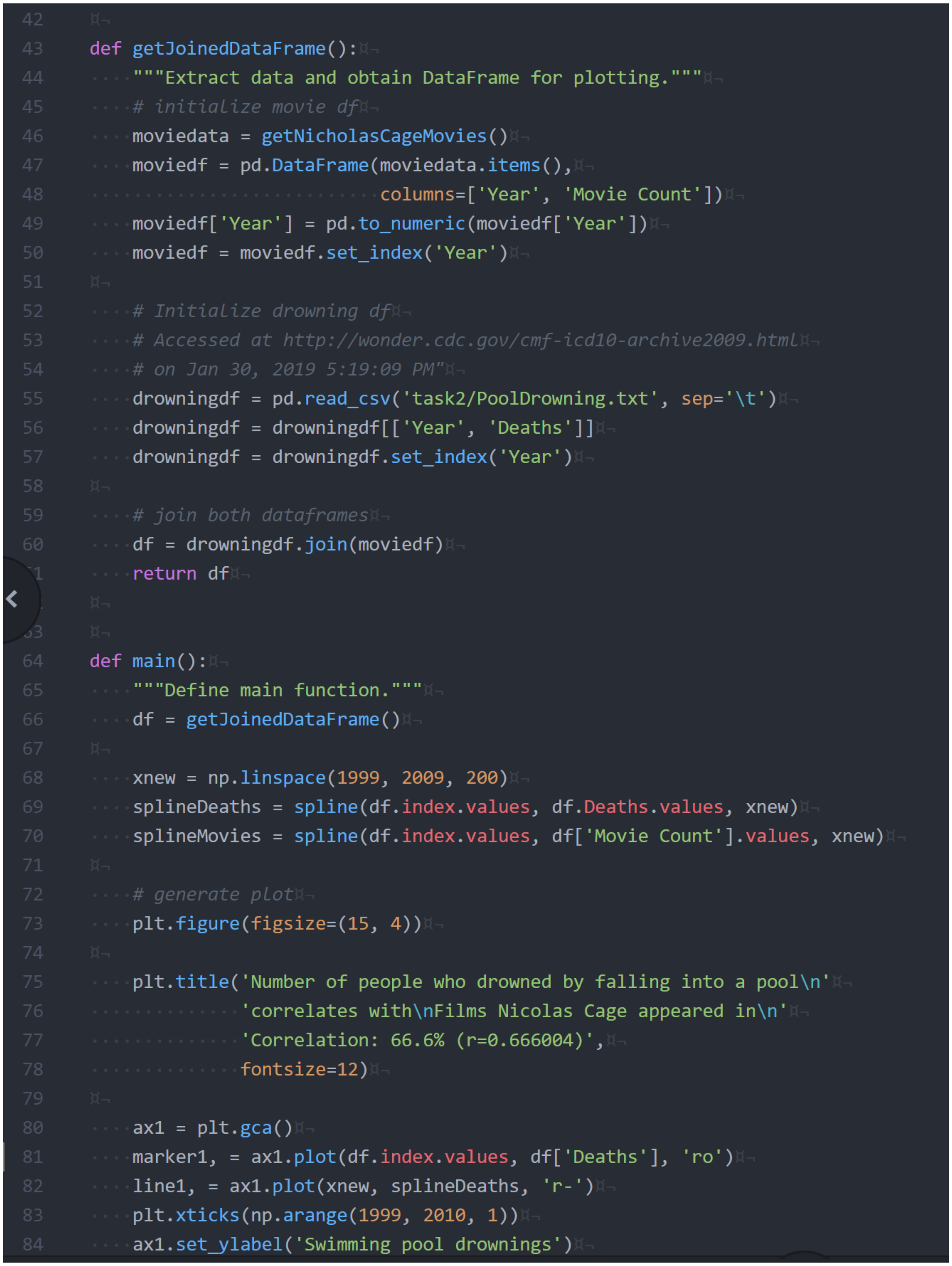
```
"""Spring 2019 COMSW 4995: Applied Machine Learning.

UNI: nl2643
Homework 1 Task 1.3

Use pandas to read 'input.txt' and test the number of rows, columns and
sum of the '2010' column.

"""

import pandas as pd

df = pd.read_csv('task1/input.txt', sep=',')
df['2010'] = pd.to_numeric(df['2010'], errors='coerce')


def test_row():
    """Check row number in df."""
    assert len(df) == 225


def test_column():
    """Check column number in df."""
    assert len(df.columns) == 32


def test_population():
    """Check sum of '2010' column in df."""
    assert round(df['2010'].sum()) == 7065
```

Task 2.1

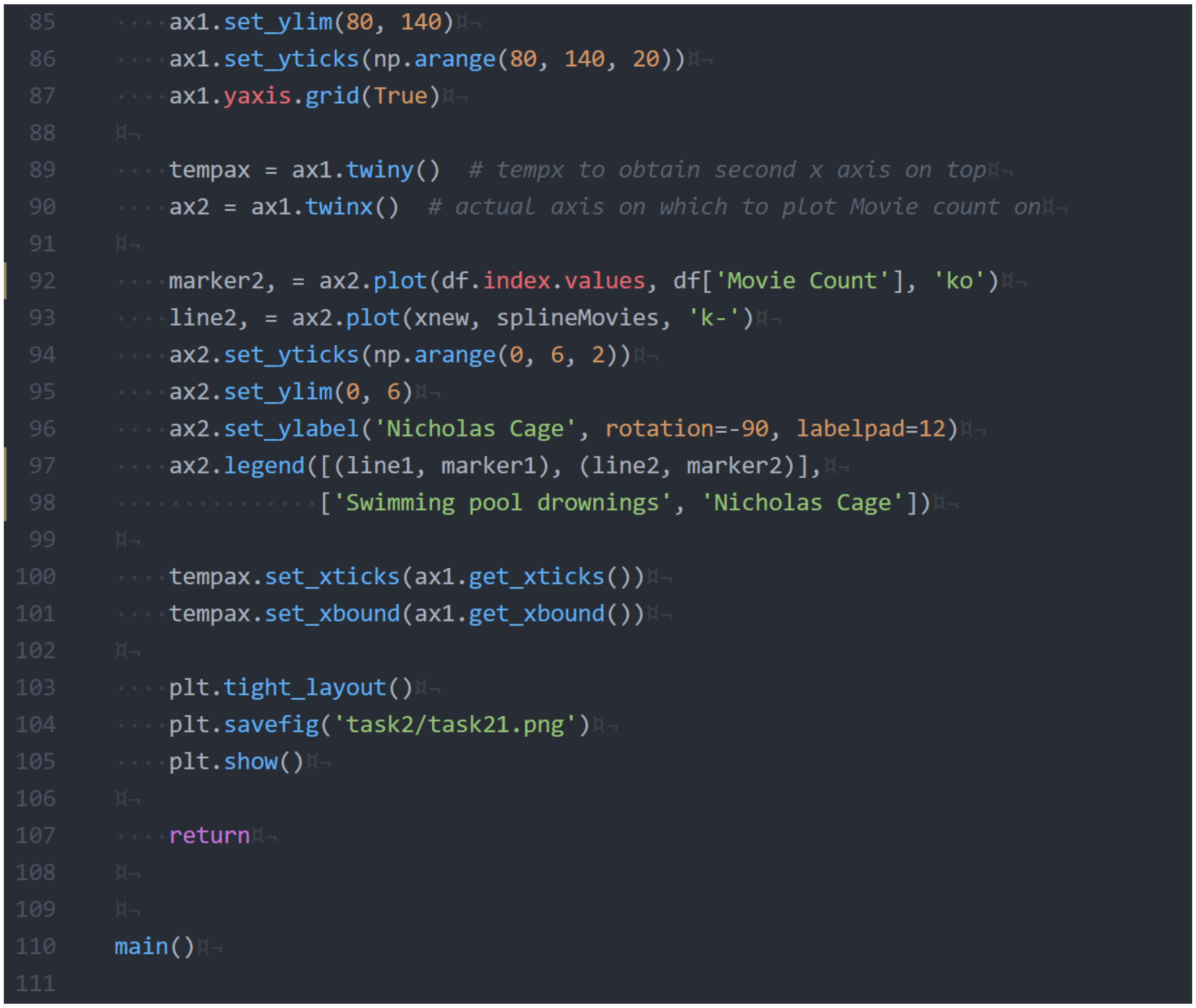
task21.py


task21.py

```python
"""Spring 2019 COMSW 4995: Applied Machine Learning.

UNI: nl2643
Homework 1 Task 2.1

Replicates plot 'Number of people who drowned by falling into a pool correlates
with Films Nicholas Cage appears in' from
www.tylervigen.com/spurious-correlations

"""

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
from scipy.interpolate import spline


def getNicholasCageMovies():
    """Fetch number of movies Nicholas Cage appears in in each year."""
    # File was retrieved from https://www.imdb.com/name/nm0000115/
    # on Jan 30 2019
    file = 'task2/NicolasCageInfo.txt'

    with open(file, "r") as file:
        text = BeautifulSoup(file, "html.parser")
        films = text.find('div', class_='filmo-category-section')
        years = films.find_all('span', class_='year_column')

    movieno = {}
    for tag in years:
        yr = tag.contents[0][2:6]
        if yr == "\n":
            pass
        elif yr in movieno:
            movieno[yr] += 1
        else:
            movieno[yr] = 1

    return movieno

```

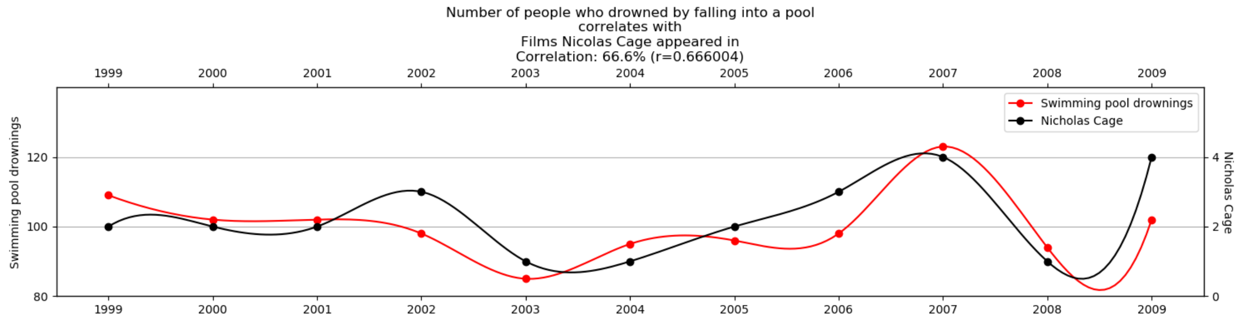


```python

def getJoinedDataFrame():
    """Extract data and obtain DataFrame for plotting."""
    # initialize movie df
    moviedata = getNicholasCageMovies()
    moviedf = pd.DataFrame(moviedata.items(),
                           columns=['Year', 'Movie Count'])
    moviedf['Year'] = pd.to_numeric(moviedf['Year'])
    moviedf = moviedf.set_index('Year')

    # Initialize drowning df
    # Accessed at http://wonder.cdc.gov/cmf-icd10-archive2009.html
    # on Jan 30, 2019 5:19:09 PM"
    drowningdf = pd.read_csv('task2/PoolDrowning.txt', sep='\t')
    drowningdf = drowningdf[['Year', 'Deaths']]
    drowningdf = drowningdf.set_index('Year')

    # join both dataframes
    df = drowningdf.join(moviedf)
    return df


def main():
    """Define main function."""
    df = getJoinedDataFrame()

    xnew = np.linspace(1999, 2009, 200)
    splineDeaths = spline(df.index.values, df.Deaths.values, xnew)
    splineMovies = spline(df.index.values, df['Movie Count'].values, xnew)

    # generate plot
    plt.figure(figsize=(15, 4))

    plt.title('Number of people who drowned by falling into a pool\n'
              'correlates with\nFilms Nicolas Cage appeared in\n'
              'Correlation: 66.6% (r=0.666004)',
              fontsize=12)

    ax1 = plt.gca()
    marker1, = ax1.plot(df.index.values, df['Deaths'], 'ro')
    line1, = ax1.plot(xnew, splineDeaths, 'r-')
    plt.xticks(np.arange(1999, 2010, 1))
    ax1.set_ylabel('Swimming pool drownings')
```



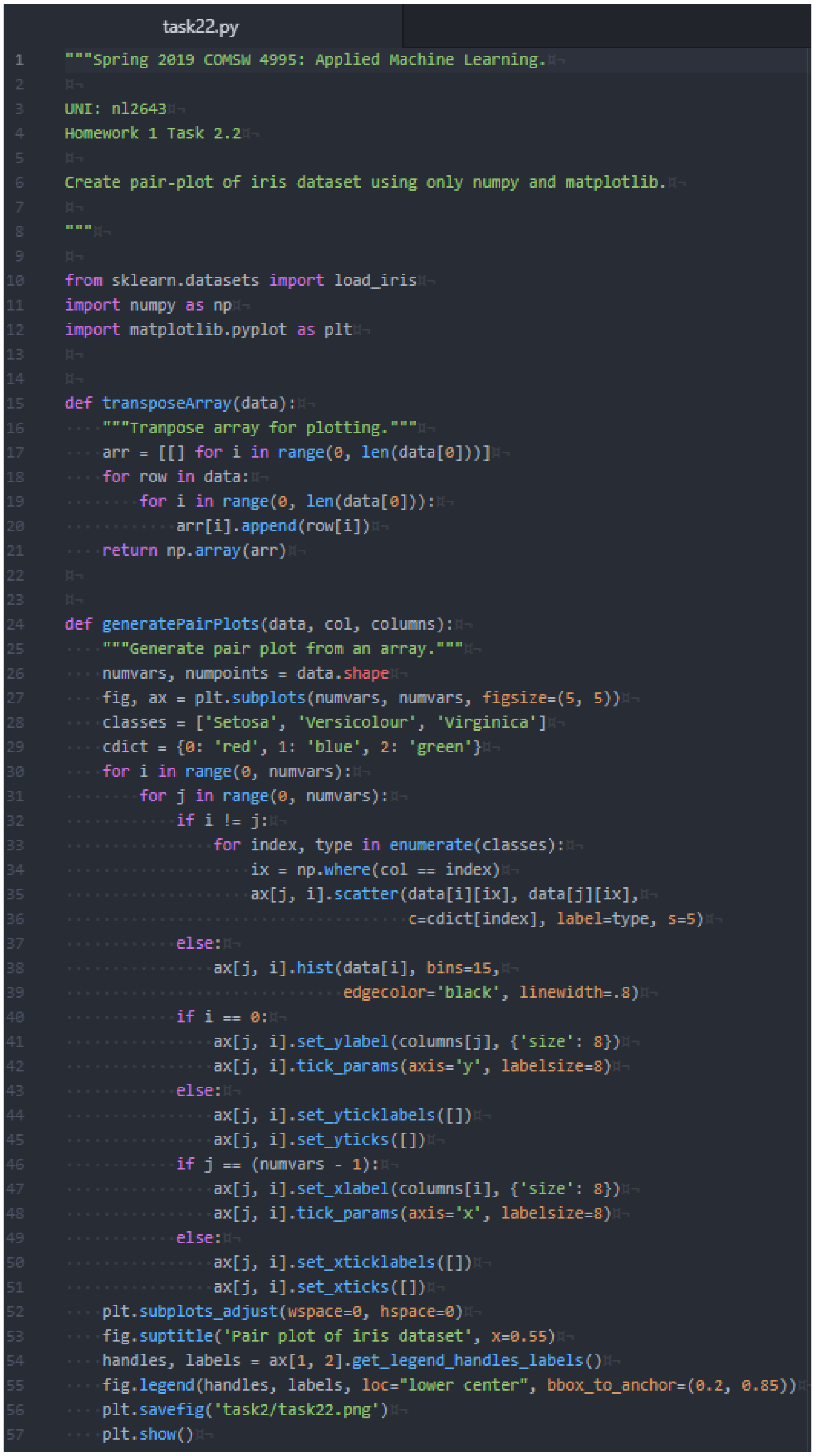
```python
    ax1.set_ylim(80, 140)
    ax1.set_yticks(np.arange(80, 140, 20))
    ax1.yaxis.grid(True)

    tempax = ax1.twiny()  # tempx to obtain second x axis on top
    ax2 = ax1.twinx()  # actual axis on which to plot Movie count on

    marker2, = ax2.plot(df.index.values, df['Movie Count'], 'ko')
    line2, = ax2.plot(xnew, splineMovies, 'k-')
    ax2.set_yticks(np.arange(0, 6, 2))
    ax2.set_ylim(0, 6)
    ax2.set_ylabel('Nicholas Cage', rotation=-90, labelpad=12)
    ax2.legend([(line1, marker1), (line2, marker2)],
               ['Swimming pool drownings', 'Nicholas Cage'])

    tempax.set_xticks(ax1.get_xticks())
    tempax.set_xbound(ax1.get_xbound())

    plt.tight_layout()
    plt.savefig('task2/task21.png')
    plt.show()

    return


main()
```

task21.png



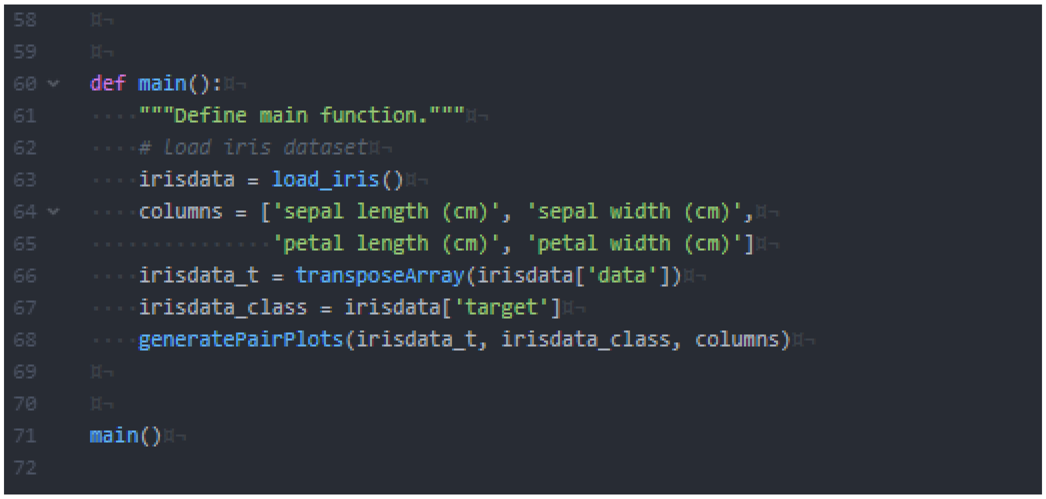
<u>Task 2.2</u>

task22.py



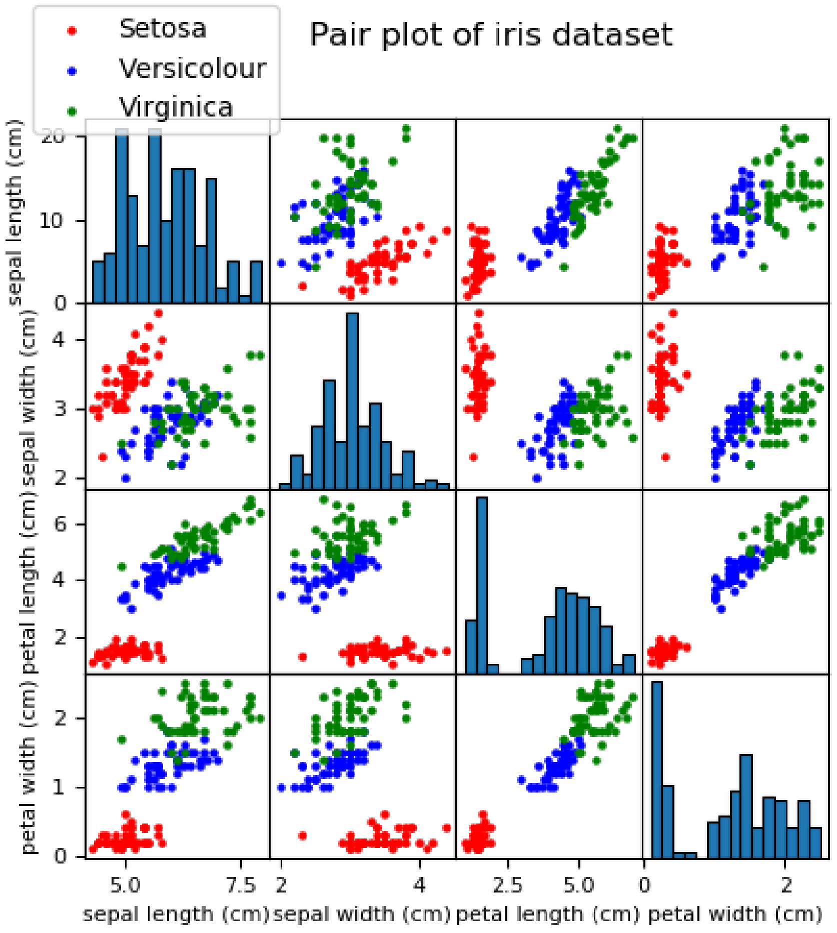
task22.py

```python
"""Spring 2019 COMSW 4995: Applied Machine Learning.

UNI: nl2643
Homework 1 Task 2.2

Create pair-plot of iris dataset using only numpy and matplotlib.

"""

from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt


def transposeArray(data):
    """Tranpose array for plotting."""
    arr = [[] for i in range(0, len(data[0]))]
    for row in data:
        for i in range(0, len(data[0])):
            arr[i].append(row[i])
    return np.array(arr)


def generatePairPlots(data, col, columns):
    """Generate pair plot from an array."""
    numvars, numpoints = data.shape
    fig, ax = plt.subplots(numvars, numvars, figsize=(5, 5))
    classes = ['Setosa', 'Versicolour', 'Virginica']
    cdict = {0: 'red', 1: 'blue', 2: 'green'}
    for i in range(0, numvars):
        for j in range(0, numvars):
            if i != j:
                for index, type in enumerate(classes):
                    ix = np.where(col == index)
                    ax[j, i].scatter(data[i][ix], data[j][ix],
                                     c=cdict[index], label=type, s=5)
            else:
                ax[j, i].hist(data[i], bins=15,
                              edgecolor='black', linewidth=.8)
            if i == 0:
                ax[j, i].set_ylabel(columns[j], {'size': 8})
                ax[j, i].tick_params(axis='y', labelsize=8)
            else:
                ax[j, i].set_yticklabels([])
                ax[j, i].set_yticks([])
            if j == (numvars - 1):
                ax[j, i].set_xlabel(columns[i], {'size': 8})
                ax[j, i].tick_params(axis='x', labelsize=8)
            else:
                ax[j, i].set_xticklabels([])
                ax[j, i].set_xticks([])
    plt.subplots_adjust(wspace=0, hspace=0)
    fig.suptitle('Pair plot of iris dataset', x=0.55)
    handles, labels = ax[1, 2].get_legend_handles_labels()
    fig.legend(handles, labels, loc="lower center", bbox_to_anchor=(0.2, 0.85))
    plt.savefig('task2/task22.png')
    plt.show()
```



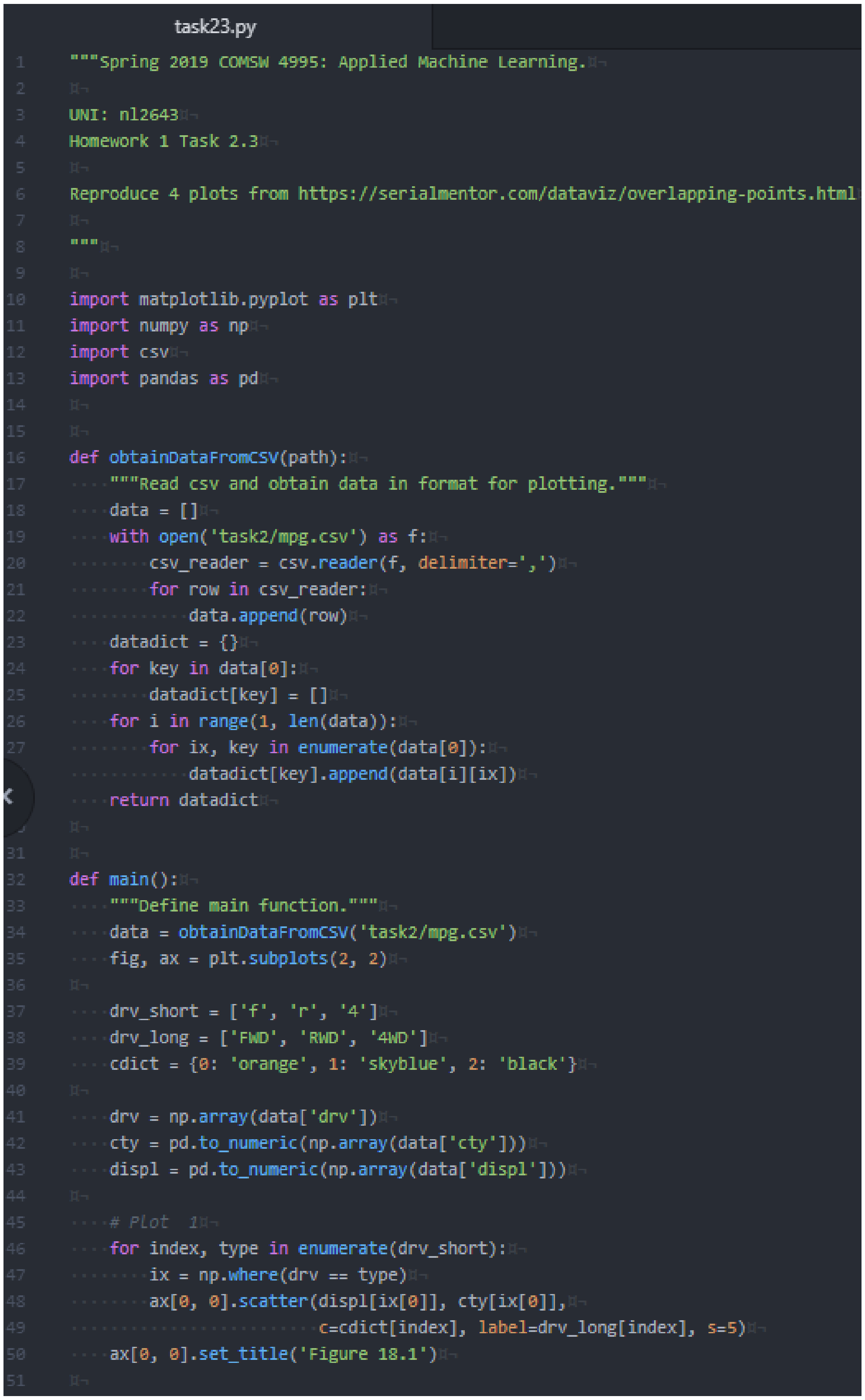
```python


def main():
    """Define main function."""
    # load iris dataset
    irisdata = load_iris()
    columns = ['sepal length (cm)', 'sepal width (cm)',
               'petal length (cm)', 'petal width (cm)']
    irisdata_t = transposeArray(irisdata['data'])
    irisdata_class = irisdata['target']
    generatePairPlots(irisdata_t, irisdata_class, columns)


main()

```

task22.png

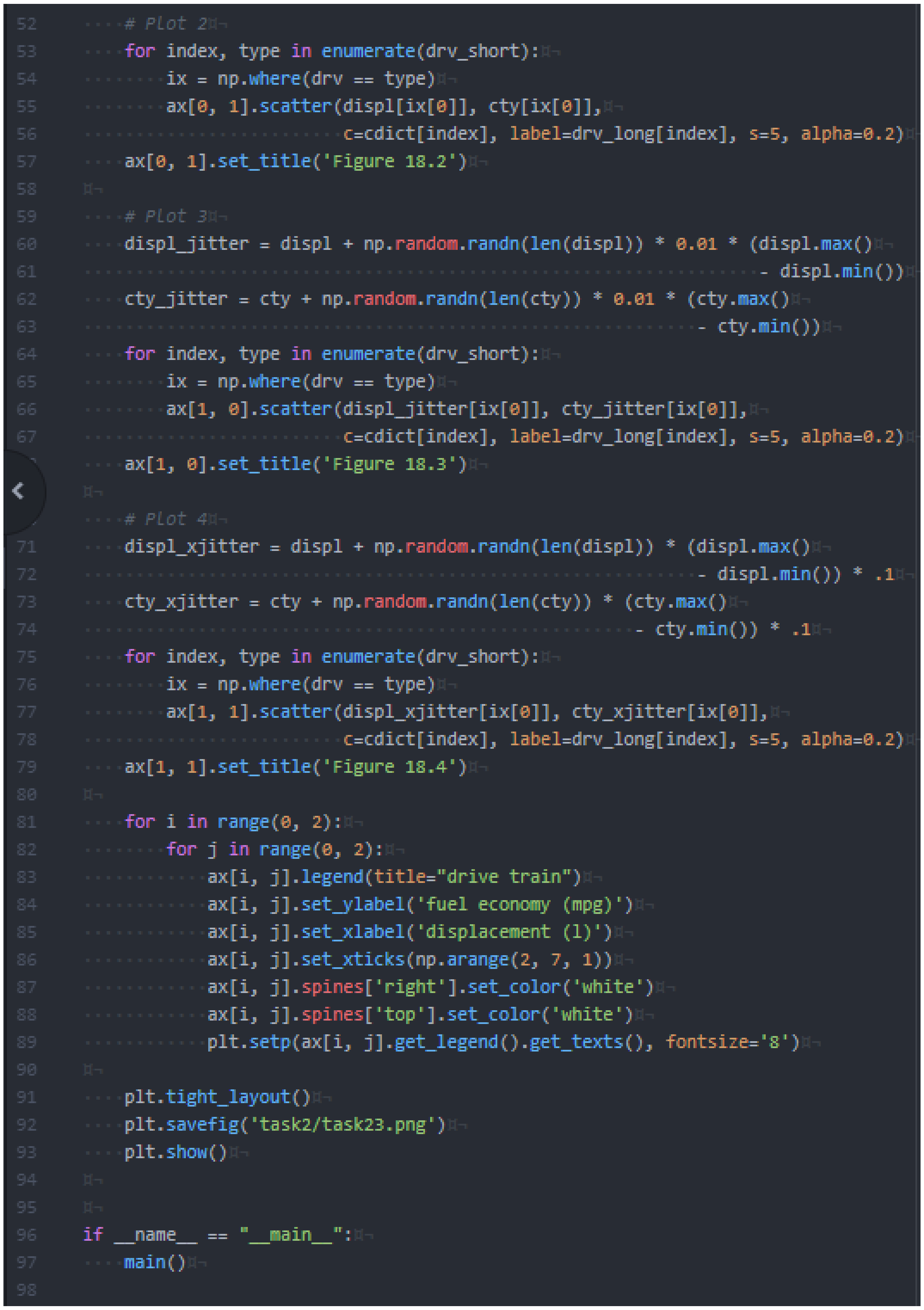


Task 2.3

task23.py



```python
"""Spring 2019 COMSW 4995: Applied Machine Learning.

UNI: nl2643
Homework 1 Task 2.3

Reproduce 4 plots from https://serialmentor.com/dataviz/overlapping-points.html

"""

import matplotlib.pyplot as plt
import numpy as np
import csv
import pandas as pd


def obtainDataFromCSV(path):
    """Read csv and obtain data in format for plotting."""
    data = []
    with open('task2/mpg.csv') as f:
        csv_reader = csv.reader(f, delimiter=',')
        for row in csv_reader:
            data.append(row)
    datadict = {}
    for key in data[0]:
        datadict[key] = []
    for i in range(1, len(data)):
        for ix, key in enumerate(data[0]):
            datadict[key].append(data[i][ix])
    return datadict


def main():
    """Define main function."""
    data = obtainDataFromCSV('task2/mpg.csv')
    fig, ax = plt.subplots(2, 2)

    drv_short = ['f', 'r', '4']
    drv_long = ['FWD', 'RWD', '4WD']
    cdict = {0: 'orange', 1: 'skyblue', 2: 'black'}

    drv = np.array(data['drv'])
    cty = pd.to_numeric(np.array(data['cty']))
    displ = pd.to_numeric(np.array(data['displ']))

    # Plot  1
    for index, type in enumerate(drv_short):
        ix = np.where(drv == type)
        ax[0, 0].scatter(displ[ix[0]], cty[ix[0]],
                         c=cdict[index], label=drv_long[index], s=5)
    ax[0, 0].set_title('Figure 18.1')

```



```python
    # Plot 2
    for index, type in enumerate(drv_short):
        ix = np.where(drv == type)
        ax[0, 1].scatter(displ[ix[0]], cty[ix[0]],
                         c=cdict[index], label=drv_long[index], s=5, alpha=0.2)
    ax[0, 1].set_title('Figure 18.2')

    # Plot 3
    displ_jitter = displ + np.random.randn(len(displ)) * 0.01 * (displ.max()
                                                                 - displ.min())
    cty_jitter = cty + np.random.randn(len(cty)) * 0.01 * (cty.max()
                                                           - cty.min())
    for index, type in enumerate(drv_short):
        ix = np.where(drv == type)
        ax[1, 0].scatter(displ_jitter[ix[0]], cty_jitter[ix[0]],
                         c=cdict[index], label=drv_long[index], s=5, alpha=0.2)
    ax[1, 0].set_title('Figure 18.3')

    # Plot 4
    displ_xjitter = displ + np.random.randn(len(displ)) * (displ.max()
                                                           - displ.min()) * .1
    cty_xjitter = cty + np.random.randn(len(cty)) * (cty.max()
                                                     - cty.min()) * .1
    for index, type in enumerate(drv_short):
        ix = np.where(drv == type)
        ax[1, 1].scatter(displ_xjitter[ix[0]], cty_xjitter[ix[0]],
                         c=cdict[index], label=drv_long[index], s=5, alpha=0.2)
    ax[1, 1].set_title('Figure 18.4')

    for i in range(0, 2):
        for j in range(0, 2):
            ax[i, j].legend(title="drive train")
            ax[i, j].set_ylabel('fuel economy (mpg)')
            ax[i, j].set_xlabel('displacement (l)')
            ax[i, j].set_xticks(np.arange(2, 7, 1))
            ax[i, j].spines['right'].set_color('white')
            ax[i, j].spines['top'].set_color('white')
            plt.setp(ax[i, j].get_legend().get_texts(), fontsize='8')

    plt.tight_layout()
    plt.savefig('task2/task23.png')
    plt.show()


if __name__ == "__main__":
    main()
```

task23.png

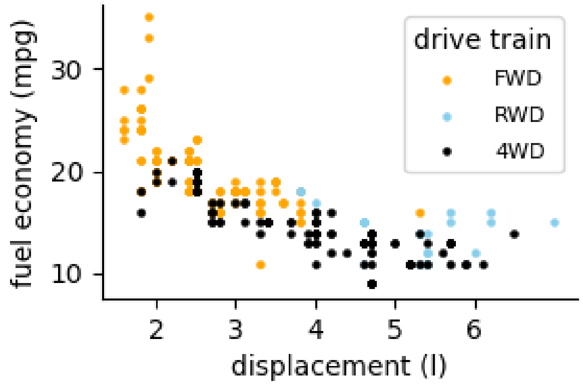


Figure 18.1

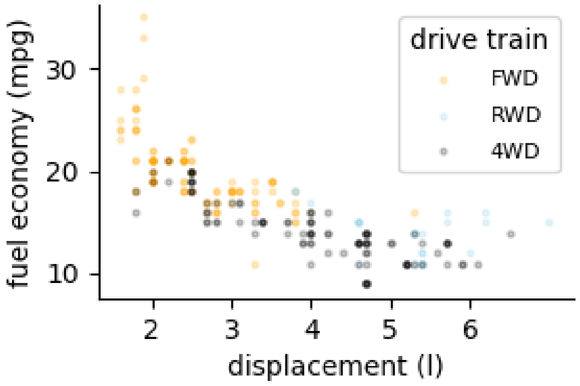


Figure 18.2

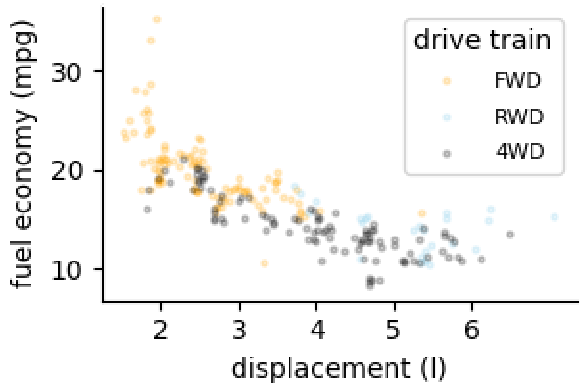


Figure 18.3

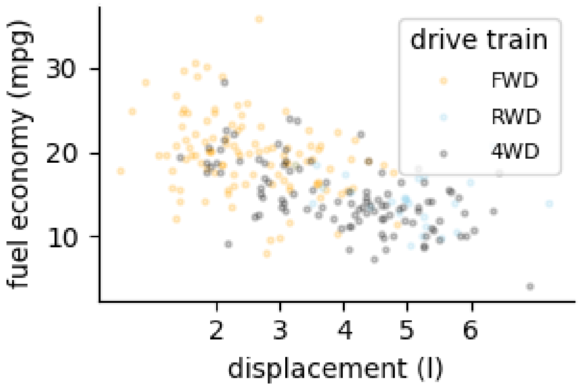


Figure 18.4