Name: - L Prathyusha

Website: - https://www.lonelyplanet.com/ (https://www.lonelyplanet.com/)

In [72]:

```
import numpy as np # linear algebra
import pandas as pd # pandas for dataframe based data processing and CSV file I/O
import requests # for http requests
from bs4 import BeautifulSoup # for html parsing and scraping
import bs4
from multiprocessing.dummy import Pool as ThreadPool

import matplotlib.pyplot as plt
import seaborn as sns
import json

sns.set_style('whitegrid')
%matplotlib inline
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [2]:

```
from bs4 import BeautifulSoup
import time
import pandas as pd
import requests
```

In [3]:

```
import requests

res = requests.get('https://www.lonelyplanet.com/')
print(res.text)
print(res.status_code)
```

```
<!DOCTYPE html><html lang="en" class="no-js"><head><link rel="dns-prefetc
h" href="https://cms.lonelyplanet.com"/><link rel="dns-prefetch" href="htt
ps://assets.lonelyplanet.com"/><link rel="dns-prefetch" href="https://lp-c
ms-production.imgix.net"/><link href="https://www.lonelyplanet.com/favico
n.ico" rel="icon" type="image/x-icon"/><meta name="theme-color" content="#
1d508d"/><link rel="preconnect" href="https://cdn.cohesionapps.com/"/><lin
k rel="preconnect" href="https://ingest.make.rvapps.io/"/><link rel="prelo
ad" as="script" href="https://cdn.cohesionapps.com/cohesion/cohesion-lates
t.min.js"/><script type="text/javascript">!function(n,e,o,r,i){
  if(!e){
    e=e||{},window.permutive=e,
    e.q=[],
    e.config=i||{},
    e.config.projectId=o,
    e.config.apiKey=r,
    e.config.environment=e.config.environment||"production";
    for(var t=["addon","identify","track","trigger","query","segment","seg
ments","ready","on","once","user","consent"],c=0;c<t.length;c++){
      var f=t[c];
```

In [4]:

```
#title
from bs4 import BeautifulSoup

page = requests.get("https://www.lonelyplanet.com/")
soup = BeautifulSoup(page.content, 'html.parser')
page_title = soup.title.text

# print the result
print(page_title)
```

```
Lonely Planet | Travel Guides & Travel Information - Lonely Planet
```

In [5]:

```
#body and head
import requests
from bs4 import BeautifulSoup

# Make a request
page = requests.get("https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_po
soup = BeautifulSoup(page.content, 'html.parser')

# Extract title of page
page_title = soup.title.text

# Extract body of page
page_body = soup.body

# Extract head of page
page_head = soup.head

# print the result
print(page_body, page_head)
```

```
<body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject m
w-editable page-List_of_countries_and_dependencies_by_population rootpage-
List_of_countries_and_dependencies_by_population skin-vector action-view s
kin-vector-legacy"><div class="noprint" id="mw-page-base"></div>
<div class="noprint" id="mw-head-base"></div>
<div class="mw-body" id="content" role="main">
<a id="top"></a>
<div id="siteNotice"><!-- CentralNotice --></div>
<div class="mw-indicators">
</div>
<h1 class="firstHeading" id="firstHeading">List of countries and dependenc
ies by population</h1>
<div id="bodyContent">
<div class="noprint" id="siteSub">From Wikipedia, the free encyclopedia</d
iv>
<div id="contentSub"></div>
<div id="contentSub2"></div>
<div id="jump-to-nav"></div>
<a class="mw-jump-link" href="#mw-head">Jump to navigation</a>
```

In [12]:

```
all_links = []
links = soup.select('a')
for ahref in links:
    text = ahref.text
    text = text.strip() if text is not None else ''

    href = ahref.get('href')
    href = href.strip() if href is not None else ''
    all_links.append({"href": href, "text": text})

print(all_links)
```

```
[{'href': '', 'text': ''}, {'href': '#mw-head', 'text': 'Jump to navigatio
n'}, {'href': '#searchInput', 'text': 'Jump to search'}, {'href': '/wiki/F
ile:World_Population.svg', 'text': ''}, {'href': '/wiki/File:World_Populat
ion.svg', 'text': ''}, {'href': '/wiki/Sovereign_state', 'text': 'sovereig
n states'}, {'href': '/wiki/Dependent_territory', 'text': 'dependent terri
tories'}, {'href': '/wiki/Country#Sovereignty', 'text': 'constituent count
ries'}, {'href': '/wiki/ISO', 'text': 'ISO'}, {'href': '/wiki/ISO_3166-1',
'text': 'ISO 3166-1'}, {'href': '/wiki/United_Kingdom', 'text': 'United Ki
ngdom'}, {'href': '/wiki/Kingdom_of_the_Netherlands', 'text': 'Kingdom of
the Netherlands'}, {'href': '/wiki/List_of_states_with_limited_recognitio
n', 'text': 'states with limited recognition'}, {'href': '/wiki/World_popu
lation', 'text': 'world population'}, {'href': '/wiki/United_Nations', 'te
xt': 'United Nations'}, {'href': '#Method', 'text': '1 Method'}, {'href':
'#Sovereign_states_and_dependencies_by_population', 'text': '2 Sovereign s
tates and dependencies by population'}, {'href': '#Notes', 'text': '3 Note
s'}, {'href': '#References', 'text': '4 References'}, {'href': '/w/index.p
hp?title=List_of_countries_and_dependencies_by_population&action=edit&sect
ion=1', 'text': 'edit'}, {'href': '/wiki/List_of_countries_and_dependencie
s_by_population_density', 'text': 'List of countries and dependencies by p
```

In [15]:

```
def ffloat(string):
    if string is None:
        return np.nan
    if type(string)==float or type(string)==np.float64:
        return string
    if type(string)==int or type(string)==np.int64:
        return string
    return fast_float(string.split(" ")[0].replace(',','').replace('%',''),
                      default=np.nan)
```

In [16]:

```
def ffloat_list(string_list):
    return list(map(ffloat,string_list))
```

In [17]:

```
def remove_multiple_spaces(string):
    if type(string)==str:
        return ' '.join(string.split())
    return string
```

In [18]:

```
response = requests.get("http://www.example.com/", timeout=240)
response.status_code
response.content
```

Out[18]:

```
b'<!doctype html>\n<html>\n<head>\n    <title>Example Domain</title>\n\n
<meta charset="utf-8" />\n    <meta http-equiv="Content-type" content="text/
html; charset=utf-8" />\n    <meta name="viewport" content="width=device-wid
th, initial-scale=1" />\n    <style type="text/css">\n    body {\n        ba
ckground-color: #f0f0f2;\n        margin: 0;\n        padding: 0;\n        f
ont-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open
Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;\n        \n    }\n
div {\n        width: 600px;\n        margin: 5em auto;\n        padding: 2e
m;\n        background-color: #fdfdff;\n        border-radius: 0.5em;\n
box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);\n    }\n    a:link, a:visited
{\n        color: #38488f;\n        text-decoration: none;\n    }\n    @medi
a (max-width: 700px) {\n        div {\n            margin: 0 auto;\n
width: auto;\n        }\n    }\n    </style>    \n</head>\n\n<body>\n<div>\n
<h1>Example Domain</h1>\n    <p>This domain is for use in illustrative examp
les in documents. You may use this\n    domain in literature without prior c
oordination or asking for permission.</p>\n    <p><a href="https://www.iana.
org/domains/example">More information...</a></p>\n</div>\n</body>\n</html>
\n'
```

In [24]:

```
from IPython.core.display import HTML
HTML("<b>Rendered HTML</b>")
response = requests.get("https://www.moneycontrol.com/india/stockpricequote/auto-2-3-wheele
page_content = BeautifulSoup(response.content, "html.parser")
HTML(str(page_content.find("h1")))
```

Out[24]:

Hero Motocorp Ltd.

In [25]:

```
response = requests.get("https://www.moneycontrol.com/india/stockpricequote/auto-2-3-wheele
content = BeautifulSoup(response.content, "html.parser")
price_div = content.find("div",attrs={"id":'b_changetext'})
HTML(str(price_div))
```

Out[25]:

None

In [26]:

```
html = '''
<table>
    <tr>
        <td>Month</td>
        <td>Price</td>
    </tr>
    <tr>
        <td>July</td>
        <td>2</td>
    </tr>
    <tr>
        <td>August</td>
        <td>4</td>
    </tr>
    <tr>
        <td>September</td>
        <td>3</td>
    </tr>
    <tr>
        <td>October</td>
        <td>2</td>
    </tr>
</table>
'''
HTML(html)
```

Out[26]:

Month	Price
July	2
August	4
September	3
October	2

In [38]:

```
def get_table_simple(table,is_table_tag=True):
    elems = table.find_all('tr') if is_table_tag else get_children(table)
    table_data = list()
    for row in elems:
        row_data = list()
        row_elems = get_children(row)
        for elem in row_elems:
            text = elem.text.strip().replace("\n","")
            text = remove_multiple_spaces(text)
            if len(text)==0:
                continue
            row_data.append(text)
        table_data.append(row_data)
    return table_data
```

In [39]:

```
html= BeautifulSoup(html,"html.parser")
get_table_simple(html)
```

Out[39]:

```
[]
```

Yahoo

In [50]:

```
!pip install pandas-datareader
```

```
Collecting pandas-datareader
  Downloading pandas_datareader-0.9.0-py3-none-any.whl (107 kB)
Requirement already satisfied: pandas>=0.23 in c:\users\prathyu lachireddy\a
naconda3\lib\site-packages (from pandas-datareader) (1.1.3)
Requirement already satisfied: requests>=2.19.0 in c:\users\prathyu lachired
dy\anaconda3\lib\site-packages (from pandas-datareader) (2.24.0)
Requirement already satisfied: lxml in c:\users\prathyu lachireddy\anaconda3
\lib\site-packages (from pandas-datareader) (4.6.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\prathyu lachireddy\a
naconda3\lib\site-packages (from pandas>=0.23->pandas-datareader) (2020.1)
Requirement already satisfied: numpy>=1.15.4 in c:\users\prathyu lachireddy
\anaconda3\lib\site-packages (from pandas>=0.23->pandas-datareader) (1.19.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\prathyu la
chireddy\anaconda3\lib\site-packages (from pandas>=0.23->pandas-datareader)
(2.8.1)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\prathyu lachireddy\anaconda3\lib\site-packages (from requests>=2.1
9.0->pandas-datareader) (1.25.11)
Requirement already satisfied: idna<3,>=2.5 in c:\users\prathyu lachireddy\a
naconda3\lib\site-packages (from requests>=2.19.0->pandas-datareader) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\prathyu lachire
ddy\anaconda3\lib\site-packages (from requests>=2.19.0->pandas-datareader)
(3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\prathyu lachir
eddy\anaconda3\lib\site-packages (from requests>=2.19.0->pandas-datareader)
(2020.6.20)
Requirement already satisfied: six>=1.5 in c:\users\prathyu lachireddy\anaco
nda3\lib\site-packages (from python-dateutil>=2.7.3->pandas>=0.23->pandas-da
tareader) (1.15.0)
Installing collected packages: pandas-datareader
Successfully installed pandas-datareader-0.9.0
```

In [51]:

```
# Import relevant packages
import yahoo_fin.stock_info as ya
from alpha_vantage.timeseries import TimeSeries
from alpha_vantage.techindicators import TechIndicators
from alpha_vantage.sectorperformance import SectorPerformances
import pandas as pd
import pandas_datareader as web
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import requests
import numpy as np

movers = ya.get_day_most_active()
movers.head()
```

Out[51]:

	Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Market Cap
0	F	Ford Motor Company	13.33	0.84	6.73	197312000.0	69246000.0	5.3209e+10
1	AAPL	Apple Inc.	125.43	-1.88	-1.48	79153000.0	102798000.0	2.093T
2	T	AT&T Inc.	30.01	0.37	1.25	74570000.0	49097000.0	2.14271e+11
3	AMC	AMC Entertainment Holdings, Inc.	12.08	-0.47	-3.75	52581000.0	94910000.0	5.439e+09
4	ABEV	Ambev S.A.	3.30	-0.05	-1.49	35297000.0	22173000.0	5.2511e+10

In [52]:

```
movers = movers[movers['% Change'] >= 0]
movers.head()
```

Out[52]:

	Symbol	Name	Price (Intraday)	Change	% Change	Volume	Avg Vol (3 month)	Market Cap	P Rati (TTM
0	F	Ford Motor Company	13.33	0.84	6.73	197312000.0	69246000.0	5.3209e+10	13.4
2	T	AT&T Inc.	30.01	0.37	1.25	74570000.0	49097000.0	2.14271e+11	Na
5	SPCE	Virgin Galactic Holdings, Inc.	21.07	1.26	6.36	50029000.0	14497000.0	5.072e+09	Na
7	GE	General Electric Company	13.23	0.17	1.30	51746000.0	76837000.0	1.16141e+11	Na
12	PLUG	Plug Power Inc.	27.89	0.42	1.53	35953000.0	38547000.0	1.585e+10	Na

In [53]:

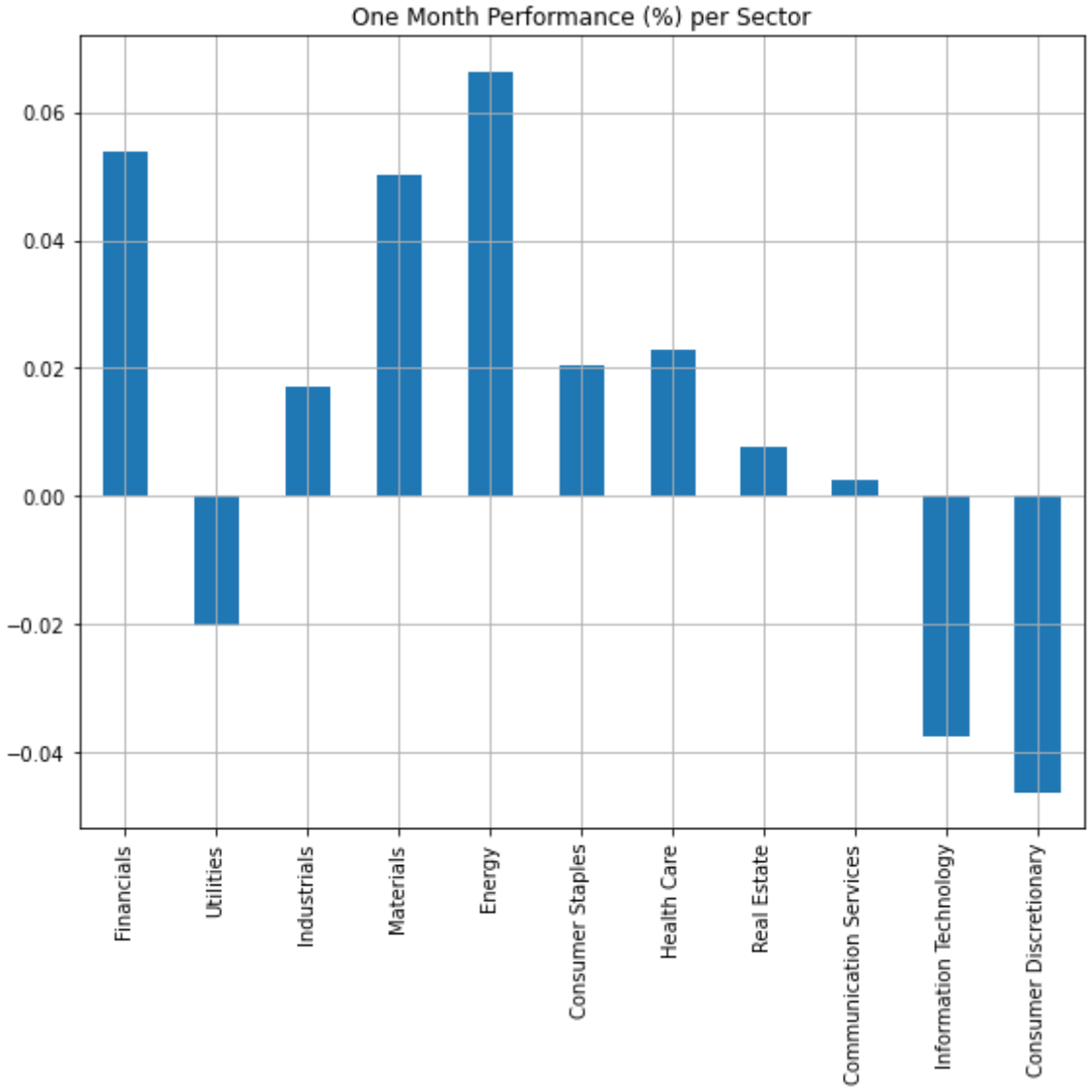
```
res = requests.get('http://www.sentdex.com/financial-analysis/?tf=30d')
soup = BeautifulSoup(res.text)
table = soup.find_all('tr')
# Initialize empty lists to store stock symbol, sentiment and mentions
stock = []
sentiment = []
mentions = []
sentiment_trend = []
# Use try and except blocks to mitigate missing data
for ticker in table:
    ticker_info = ticker.find_all('td')

    try:
        stock.append(ticker_info[0].get_text())
    except:
        stock.append(None)
    try:
        sentiment.append(ticker_info[3].get_text())
    except:
        sentiment.append(None)
    try:
        mentions.append(ticker_info[2].get_text())
    except:
        mentions.append(None)
    try:
        if (ticker_info[4].find('span',{"class":"glyphicon  glyphicon-chevron-up"})):
            sentiment_trend.append('up')
        else:
            sentiment_trend.append('down')
    except:
        sentiment_trend.append(None)
```

In [55]:

```
# Checking sector performances
sp = SectorPerformances(key='0E66O7ZP6W7A1LC9', output_format='pandas')
plt.figure(figsize=(8,8))
data, meta_data = sp.get_sector()
print(meta_data)
data['Rank D: Month Performance'].plot(kind='bar')
plt.title('One Month Performance (%) per Sector')
plt.tight_layout()
plt.grid()
plt.show()
```

```
{'Information': 'US Sector Performance (realtime & historical)', 'Last Refre
shed': '2021-05-23 10:38:16 US/Eastern'}
```



In [57]:

```
# How to obtain technical indicators for a stock of choice
# RSI
t_rsi = TechIndicators(key='0E66O7ZP6W7A1LC9',output_format='pandas')
data_rsi, meta_data_rsi = t_rsi.get_rsi(symbol='AMD', interval='daily',time_period = 9,
                        series_type='open')
# Bollinger bands
t_bbands = TechIndicators(key='0E66O7ZP6W7A1LC9',output_format='pandas')
data_bbands, meta_data_bb = t_bbands.get_bbands(symbol='AMD', interval='daily', series_type
```

In [61]:

```
!pip install sklearn
```

```
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Requirement already satisfied: scikit-learn in c:\users\prathyu lachireddy\a
naconda3\lib\site-packages (from sklearn) (0.24.2)
Requirement already satisfied: numpy>=1.13.3 in c:\users\prathyu lachireddy
\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.19.2)
Requirement already satisfied: scipy>=0.19.1 in c:\users\prathyu lachireddy
\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.5.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\prathyu lach
ireddy\anaconda3\lib\site-packages (from scikit-learn->sklearn) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\prathyu lachireddy\a
naconda3\lib\site-packages (from scikit-learn->sklearn) (0.17.0)
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py): started
  Building wheel for sklearn (setup.py): finished with status 'done'
  Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.whl size=
1321 sha256=d748bb80edb155152367e9017a1686036d3cc59a207059c6da1f17dc72357fbc
  Stored in directory: c:\users\prathyu lachireddy\appdata\local\pip\cache\w
heels\22\0b\40\fd3f795caaa1fb4c6cb738bc1f56100be1e57da95849bfc897
Successfully built sklearn
Installing collected packages: sklearn
Successfully installed sklearn-0.0
```

In [ ]:

```
# Feature selection using a Random forest regressor
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=0)
feat = SelectFromModel(RandomForestRegressor(n_estimators=100,random_state=0,n_jobs=-1))
feat.fit(X_train,y_train)
X_train.columns[feat.get_support()]
```

In [71]:

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import GridSearchCV, cross_val_score
import sklearn
from sklearn.metrics import median_absolute_error, mean_squared_error
# Split data into training and test
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg, y_reg, test_size
# Use a 10-fold cross validation to determine optimal parameters for a ridge regression
ridge = Ridge()
alphas = [1e-15,1e-8,1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,0,1,5,10,20,30,40,45,50,55,100]
params = {'alpha': alphas}
ridge_regressor = GridSearchCV(ridge,params, scoring='neg_mean_squared_error',cv=10)
ridge_regressor.fit(X_reg,y_reg)
print(ridge_regressor.best_score_)
print(ridge_regressor.best_params_)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-71-7fcd4be34094> in <module>
      4 from sklearn.metrics import median_absolute_error, mean_squared_erro
r
      5 # Split data into training and test
----> 6 X_train_reg = train_test_split(X_reg, y_reg, test_size=0.2, random_s
tate=0)
      7 # Use a 10-fold cross validation to determine optimal parameters for
a ridge regression
      8 ridge = Ridge()

NameError: name 'X_reg' is not defined
```

In [65]:

```
regr = Ridge(alpha=1e-15)
regr.fit(X_train_reg, y_train_reg)
y_pred = regr.predict(X_test_reg)
y_pred_train = regr.predict(X_train_reg)
print(f'R^2 value for test set is {regr.score(X_test_reg,y_test_reg)}')
print(f'Mean squared error is {mean_squared_error(y_test_reg,y_pred)}')
plt.scatter(df_updated['Open-Close'][1:],df_updated['Diff_Close'][1:],c='k')
plt.plot(df_updated['Open-Close'][1:], (regr.coef_[0] * df_updated['Open-Close'][1:] + regr
plt.xlabel('Open-Close')
plt.ylabel('Diff-Close')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-65-cdc52eeaaa3c> in <module>
      1 regr = Ridge(alpha=1e-15)
----> 2 regr.fit(X_train_reg, y_train_reg)
      3 y_pred = regr.predict(X_test_reg)
      4 y_pred_train = regr.predict(X_train_reg)
      5 print(f'R^2 value for test set is {regr.score(X_test_reg,y_test_re
g)}')

NameError: name 'X_train_reg' is not defined
```

In [70]:

```
def predict_range(X,y,model,conf=2.58):

    from numpy import sum as arraysum

    # Obtain predictions
    yhat = model.predict(X)

    # Compute standard deviation
    sum_errs = arraysum((y - yhat)**2)
    stdev = np.sqrt(1/(len(y)-2) * sum_errs)

    # Prediction interval (default confidence: 99%)
    interval = conf * stdev

    lower = []
    upper = []
    for i in yhat:
        lower.append(i-interval)
        upper.append(i+interval)


    return lower, upper, interval
```

In [ ]:

```

```