Linear_Regression_Trump_vs_Biden

November 30, 2020

[ ]:
```
![ -e 'battleground-state-changes.csv' ] || wget https://raw.githubusercontent.
 ↪com/alex/nyt-2020-election-scraper/master/battleground-state-changes.csv
#![ -e 'zip.train' ] || wget https://web.stanford.edu/~hastie/ElemStatLearn/
 ↪datasets/zip.train.gz && gzip -d zip.train.gz
#![ -e 'zip.test' ] || wget https://web.stanford.edu/~hastie/ElemStatLearn/
 ↪datasets/zip.test.gz && gzip -d zip.test.gz
```

[ ]:
```
import os
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from enum import Enum
```

[ ]:
```
class Colors(Enum):
    blue = '#0A85FF'
    darkblue = '#00264D'
    green = '#99CC00'
    darkgreen = '#739900'
```

[ ]:
```
battleground_path = './battleground-state-changes.csv'
df = pd.read_csv(battleground_path)
```

[ ]:
```
# Restructure data frame for easier access to Biden and Trump votes

def get_candidate_votes(row):
    if row['leading_candidate_name'] == 'Biden':
        row['biden_votes'] = row['leading_candidate_votes']
        row['trump_votes'] = row['trailing_candidate_votes']
    elif row['leading_candidate_name'] == 'Trump':
        row['biden_votes'] = row['trailing_candidate_votes']
        row['trump_votes'] = row['leading_candidate_votes']
    return row

df = df.apply(get_candidate_votes, axis=1)
```

[ ]:
```
df
```

[ ]:
```
                      state  … trump_votes
0           Alaska (EV: 3)  …      189543
1           Alaska (EV: 3)  …      189484
2           Alaska (EV: 3)  …      189457
3           Alaska (EV: 3)  …      189457
4           Alaska (EV: 3)  …      185769
..                       …  …         …
755  Pennsylvania (EV: 20)  …     2977987
756  Pennsylvania (EV: 20)  …     2976718
757  Pennsylvania (EV: 20)  …     2976682
758  Pennsylvania (EV: 20)  …     2970396
759  Pennsylvania (EV: 20)  …     2969504

[760 rows x 22 columns]
```

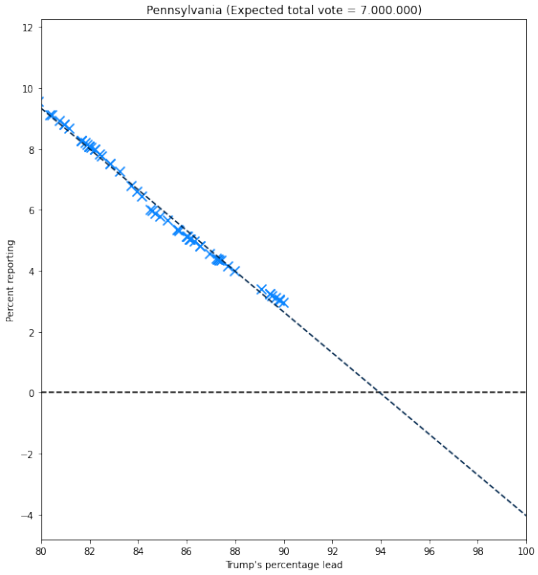
[ ]:
```python
class LinearRegression():
  def __init__(self, w=0, b=0):
    self.w = w
    self.b = b

  # def fit(X,y):
  #   w = 0
  #   b = 0
  #   return LinearRegression(w,b)

  def fit(self, X, y):
    X_u = np.hstack([X, np.ones_like(X[:, 0]).reshape([-1, 1])])
    wb = np.linalg.pinv(X_u) @ y
    # y   X_u @ wb = X @ w + b
    w, b = wb[:-1], wb[-1]
    self.w = w
    self.b = b
    return self

  def predict(self, x):
    """
    Define the line here.
    Map x to it's y coordinates. The set {(x,y) | x from input, y from output}
    should define your line.
    In this case x will be percentages reported like
    np.linspace(0.94, 1, 100).reshape([-1,1]) from 94% to 100%
    in Georgia
    """

    return x @ self.w + self.b
```

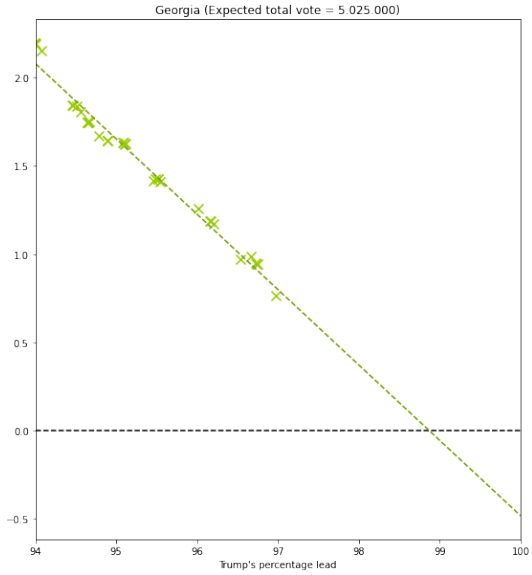
[ ]:
```
#Pennsylvania
df_py = df[df['state'] == 'Pennsylvania (EV: 20)']
df_py['percent_reporting'] = df_py['total_votes_count'] / int(7e+6)
df_py['trump_lead'] = (df_py['trump_votes'] - df_py['biden_votes']) /␣
↪df_py['total_votes_count']
df_py = df_py.sort_values('percent_reporting', axis=0, ascending=False)

# Only view votes that were counted before 90% were reported
df_py = df_py[df_py['percent_reporting'] <= 0.9]

X = df_py['percent_reporting'].to_numpy().reshape([-1, 1])
y = df_py['trump_lead'].to_numpy()

# Fit X to y using Linear Regression
linear_py = LinearRegression().fit(X, y)



#Georgia
df_ga = df[df['state'] == 'Georgia (EV: 16)']
df_ga['percent_reporting'] = df_ga['total_votes_count'] / int(5.025e+6)
df_ga['trump_lead'] = (df_ga['trump_votes'] - df_ga['biden_votes']) /␣
↪df_ga['total_votes_count']
df_ga = df_ga.sort_values('percent_reporting', axis=0, ascending=False)

# Only view votes that were counted before 97% were reported
df_ga = df_ga[df_ga['percent_reporting'] <= 0.97]

X = df_ga['percent_reporting'].to_numpy().reshape([-1, 1])
y = df_ga['trump_lead'].to_numpy()

# Fit X to y using Linear Regression
linear_ga = LinearRegression().fit(X, y)

fig, ax = plt.subplots(1, 2, figsize=(20, 10))
ax[0].scatter(df_py['percent_reporting'] * 100, df_py['trump_lead'] * 100,␣
↪marker='x', s=100, color=Colors.blue.value)
x = np.linspace(0.8, 1, 100).reshape([-1,1])
y = linear_py.predict(x)
ax[0].set_xlim([85, 100])
ax[0].set_xticks(range(80, 101, 2))
ax[0].plot(x * 100, y * 100, linestyle='--', color=Colors.darkblue.value)
ax[0].hlines(y=0, xmin=80, xmax=100, linestyle='--', color='black')
ax[0].set_title('Pennsylvania (Expected total vote = 7.000.000)')

ax[1].scatter(df_ga['percent_reporting'] * 100, df_ga['trump_lead'] * 100,␣
↪marker='x', s=100, color=Colors.green.value)
x = np.linspace(0.94, 1, 100).reshape([-1,1])
```

```
y = linear_ga.predict(x)
ax[1].set_xlim([94, 100])
ax[1].set_xticks(range(94, 101, 1))
ax[1].plot(x * 100, y * 100, linestyle='--', color=Colors.darkgreen.value)
ax[1].hlines(y=0, xmin=94, xmax=100, linestyle='--', color='black')
ax[1].set_title('Georgia (Expected total vote = 5.025.000)')

ax[0].set_ylabel("Percent reporting")
ax[0].set_xlabel("Trump's percentage lead")
ax[1].set_xlabel("Trump's percentage lead")
# plt.savefig('trump_lead.svg', transparent=True)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
Text(0.5, 0, "Trump's percentage lead")
```





At what percentage of reported votes will Biden lead over Trump?

```
[ ]: for state, model in (('py', linear_py), ('ga', linear_ga)):
       print(state + ':', -model.b / model.w)
```

```
py: [0.93957541]
ga: [0.98868753]
```

Pennsylvania — at 94%

Georgia — at 99%