hw1q3

March 4, 2021

0.0.1 Liam Jackson

0.0.2 BE700 ML with Andy Fan

0.0.3 HW1q3

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# %matplotlib inline
```

1. Loading Dataset

[2]:
```
data_df = pd.read_csv('data.csv')
num_cols = data_df.columns.size
print(f"There are {num_cols} columns in the raw dataframe. There is one Unnamed␣
 ↪column, the 33rd column, which contains no data.")
for ind, name in enumerate(data_df.columns):
    print(f"Column number: {ind +1}, Column name: {name}")
```

```
There are 33 columns in the raw dataframe. There is one Unnamed column, the 33rd
column, which contains no data.
Column number: 1, Column name: id
Column number: 2, Column name: diagnosis
Column number: 3, Column name: radius_mean
Column number: 4, Column name: texture_mean
Column number: 5, Column name: perimeter_mean
Column number: 6, Column name: area_mean
Column number: 7, Column name: smoothness_mean
Column number: 8, Column name: compactness_mean
Column number: 9, Column name: concavity_mean
Column number: 10, Column name: concave points_mean
Column number: 11, Column name: symmetry_mean
Column number: 12, Column name: fractal_dimension_mean
Column number: 13, Column name: radius_se
Column number: 14, Column name: texture_se
Column number: 15, Column name: perimeter_se
Column number: 16, Column name: area_se
Column number: 17, Column name: smoothness_se
```

```
Column number: 18, Column name: compactness_se
Column number: 19, Column name: concavity_se
Column number: 20, Column name: concave points_se
Column number: 21, Column name: symmetry_se
Column number: 22, Column name: fractal_dimension_se
Column number: 23, Column name: radius_worst
Column number: 24, Column name: texture_worst
Column number: 25, Column name: perimeter_worst
Column number: 26, Column name: area_worst
Column number: 27, Column name: smoothness_worst
Column number: 28, Column name: compactness_worst
Column number: 29, Column name: concavity_worst
Column number: 30, Column name: concave points_worst
Column number: 31, Column name: symmetry_worst
Column number: 32, Column name: fractal_dimension_worst
Column number: 33, Column name: Unnamed: 32
```

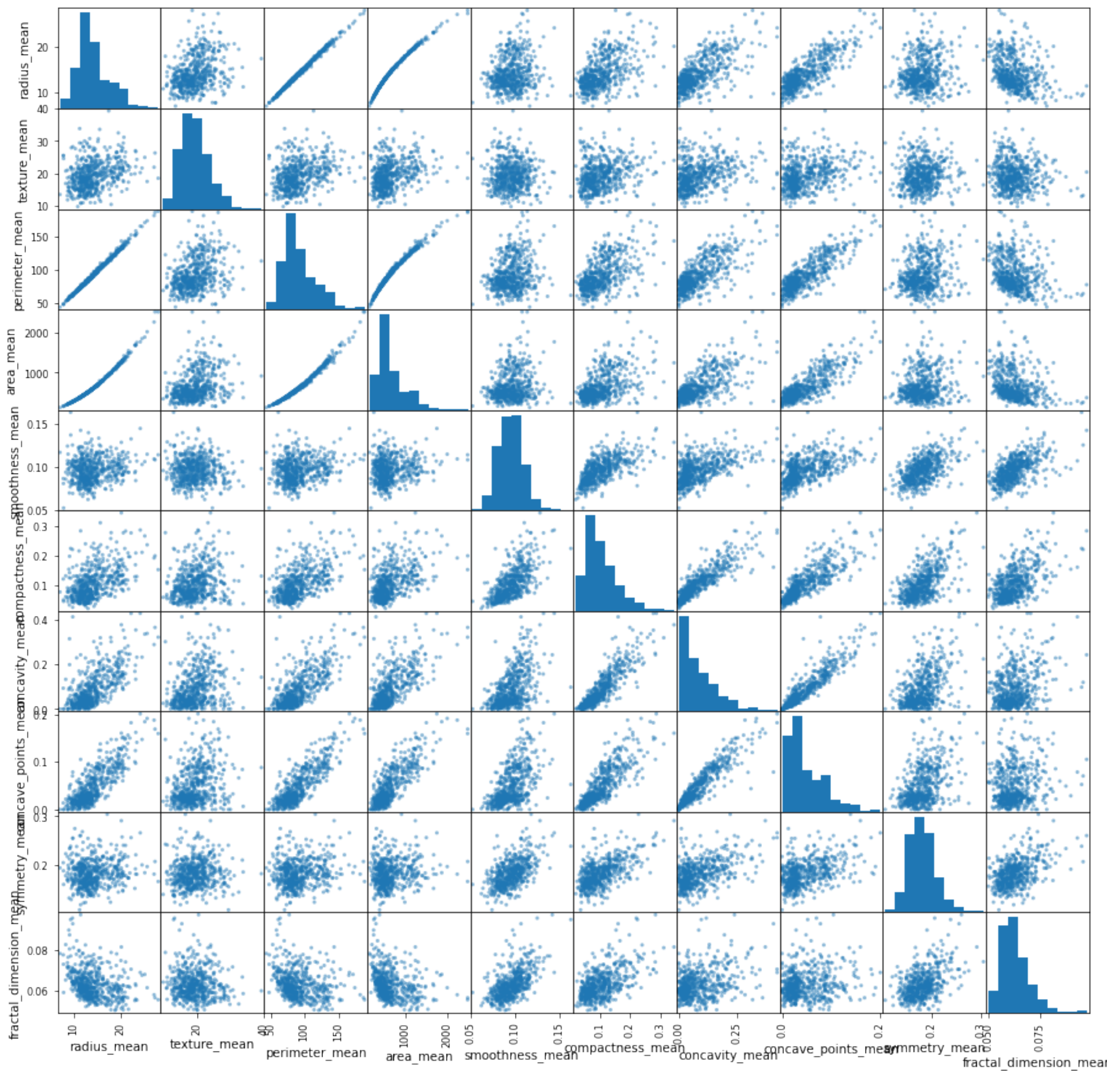
2. Generating a matrix scatter plot

[3]:
```
mean_cols = [mean_col for mean_col in data_df.columns if 'mean' in mean_col]
mean_cols = data_df.filter(regex = 'mean').columns
print(f"Just verifying there are {len(mean_cols)} columns with 'mean' in the␣
↪column name")
```

```
Just verifying there are 10 columns with 'mean' in the column name
```

[4]:
```
means_df = data_df.filter(items = mean_cols)
means_df.columns = means_df.columns.str.replace(' ', '_')
```

[5]:
```
mat_plot = pd.plotting.scatter_matrix(means_df, figsize = (15, 15))
```



3. Calculations/Statistics

[6]:
```
num_ben = data_df['diagnosis'].value_counts()['B']
num_mal = data_df['diagnosis'].value_counts()['M']
print(f"There are {num_ben} Benign occurrences and {num_mal} Malignant␣
↪occurrences.")
```

```
There are 357 Benign occurrences and 212 Malignant occurrences.
```

[7]:
```
stats_df = means_df.describe().loc[["mean", "std"], :]
print("Statistics for both diagnoses: ")
stats_df
```

```
Statistics for both diagnoses:
```

```
[7]:       radius_mean  texture_mean  perimeter_mean   area_mean  smoothness_mean  \
mean    14.127292     19.289649       91.969033  654.889104         0.096360
std      3.524049      4.301036       24.298981  351.914129         0.014064

       compactness_mean  concavity_mean  concave_points_mean  symmetry_mean  \
mean          0.104341        0.088799             0.048919       0.181162
std           0.052813        0.079720             0.038803       0.027414

       fractal_dimension_mean
mean                 0.062798
std                  0.007060
```

```
[8]: ben_df = data_df[data_df['diagnosis'] == 'B'].filter(items = mean_cols)
ben_stats_df = ben_df.describe().loc[["mean", "std"], :]
print("Benign Occurrences Statistics: ")
ben_stats_df
```

```
Benign Occurrences Statistics:
```

```
[8]:       radius_mean  texture_mean  perimeter_mean   area_mean  smoothness_mean  \
mean    12.146524     17.914762       78.075406  462.790196         0.092478
std      1.780512      3.995125       11.807438  134.287118         0.013446

       compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
mean          0.080085        0.046058             0.025717       0.174186
std           0.033750        0.043442             0.015909       0.024807

       fractal_dimension_mean
mean                 0.062867
std                  0.006747
```

```
[9]: mal_df = data_df[data_df['diagnosis'] == 'M'].filter(items = mean_cols)
mal_stats_df = mal_df.describe().loc[["mean", "std"], :]
print("Malignant Occurrences Statistics: ")
mal_stats_df
```

```
Malignant Occurrences Statistics:
```

```
[9]:       radius_mean  texture_mean  perimeter_mean   area_mean  smoothness_mean  \
mean    17.462830     21.604906      115.365377  978.376415         0.102898
std      3.203971      3.779470       21.854653  367.937978         0.012608

       compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
mean          0.145188        0.160775             0.087990       0.192909
std           0.053987        0.075019             0.034374       0.027638

       fractal_dimension_mean
```

```
mean                    0.062680
std                     0.007573
```

[10]:
```
num_ben_rad15 = ben_df[ben_df['radius_mean'] >= 15].shape[0]
per_ben_rad15 = round(100 * (num_ben_rad15 / num_ben), 2)
print(f"{per_ben_rad15} % of Benign occurrences have a cell radius of at least␣
↪15")
```

```
3.64 % of Benign occurrences have a cell radius of at least 15
```

4. Building OLS Model to predict area (y) given radius (x)

[11]:
```
xy_df = data_df[['radius_mean', 'area_mean']].sort_values('radius_mean')
xy_df
```

[11]:
```
     radius_mean  area_mean
101        6.981      143.5
539        7.691      170.4
538        7.729      178.8
568        7.760      181.0
46         8.196      201.9
..           …          …
82        25.220     1878.0
352       25.730     2010.0
180       27.220     2250.0
461       27.420     2501.0
212       28.110     2499.0

[569 rows x 2 columns]
```

[12]:
```
max_poly_ord = 2
poly_ord_range = list(range(1, max_poly_ord + 1))

ols_all_models_dict = {}

for model_ord in poly_ord_range:
    model_dict = {}
    model_df = xy_df.copy()
    x = model_df['radius_mean']
    y_r = model_df['area_mean']

    X = np.zeros([len(x), model_ord + 1])

    for X_col in list(range(0, X.shape[1])):
        X[:,X_col] = x ** X_col
    Xt = np.transpose(X)
    XtX = np.matmul(Xt, X)
    XtXinv = np.linalg.inv(XtX)
```

```
    XtXinvXt = np.matmul(XtXinv, Xt)
    beta = np.matmul(XtXinvXt, y_r)
    beta_flip = np.flipud(beta)

    y_m = np.polyval(beta_flip, x)
    res = y_r.subtract(y_m)
    res_sq = abs(res) ** 2

    model_df['y_m'] = y_m
    model_df['res'] = res
    model_df['res_sq'] = res_sq

    res_sq_sum = sum(res_sq)
    MSE = res_sq_sum / len(x)

    model_dict['beta_flip'] = beta_flip
    model_dict['model_df'] = model_df
    model_dict['res_sq_sum'] = res_sq_sum
    model_dict['MSE'] = MSE

    model_key = "p = " + str(model_ord)
    ols_all_models_dict[model_key] = model_dict
```

[13]:
```
ols_all_models_dict
```

[13]:
```
{'p = 1': {'beta_flip': array([  98.59821922, -738.0367042 ]),
  'model_df':      radius_mean  area_mean          y_m          res
res_sq
  101        6.981      143.5   -49.722536  193.222536   37334.948362
  539        7.691      170.4    20.282200  150.117800   22535.353941
  538        7.729      178.8    24.028932  154.771068   23954.083453
  568        7.760      181.0    27.085477  153.914523   23689.680417
  46         8.196      201.9    70.074300  131.825700   17378.015051
  ..           …          …          …          …            …
  82        25.220     1878.0  1748.610384  129.389616   16741.672622
  352       25.730     2010.0  1798.895476  211.104524   44565.119965
  180       27.220     2250.0  1945.806823  304.193177   92533.489030
  461       27.420     2501.0  1965.526467  535.473533  286731.904882
  212       28.110     2499.0  2033.559238  465.440762  216635.102985

  [569 rows x 5 columns],
  'res_sq_sum': 1767428.9562542248,
  'MSE': 3106.2020320812385},
 'p = 2': {'beta_flip': array([  3.10992516,   0.43684601, -10.5164038 ]),
  'model_df':      radius_mean  area_mean          y_m          res        res_sq
  101        6.981      143.5   144.093434   -0.593434      0.352164
```

```
539         7.691      170.4   176.800058   -6.400058     40.960745
538         7.729      178.8   178.638950    0.161050      0.025937
568         7.760      181.0   180.145751    0.854249      0.729742
46          8.196      201.9   201.971393   -0.071393      0.005097
..            …          …          …           …            …
82         25.220     1878.0  1978.563778 -100.563778  10113.073432
352        25.730     2010.0  2059.596420  -49.596420   2459.804862
180        27.220     2250.0  2305.606421  -55.606421   3092.074085
461        27.420     2501.0  2339.679053  161.320947  26024.448049
212        28.110     2499.0  2459.139436   39.860564   1588.864558

[569 rows x 5 columns],
'res_sq_sum': 123097.70230710595,
'MSE': 216.34042584728638}}
```

[14]:
```
print(f"The Linear (p = 1) model coefficients are: {ols_all_models_dict['p =␣
↪1']['beta_flip']}")
print("The Linear (p = 1) model residuals are: ")
print(ols_all_models_dict['p = 1']['model_df']['res'])
```

```
The Linear (p = 1) model coefficients are: [  98.59821922 -738.0367042 ]
The Linear (p = 1) model residuals are:
101    193.222536
539    150.117800
538    154.771068
568    153.914523
46     131.825700
          …
82     129.389616
352    211.104524
180    304.193177
461    535.473533
212    465.440762
Name: res, Length: 569, dtype: float64
```

[15]:
```
print(f"The Quadratic (p = 2) model coefficients are: {ols_all_models_dict['p =␣
↪2']['beta_flip']}")
print(f"The Quadratic (p = 2) model residuals are: ")
print(ols_all_models_dict['p = 2']['model_df']['res'])
```

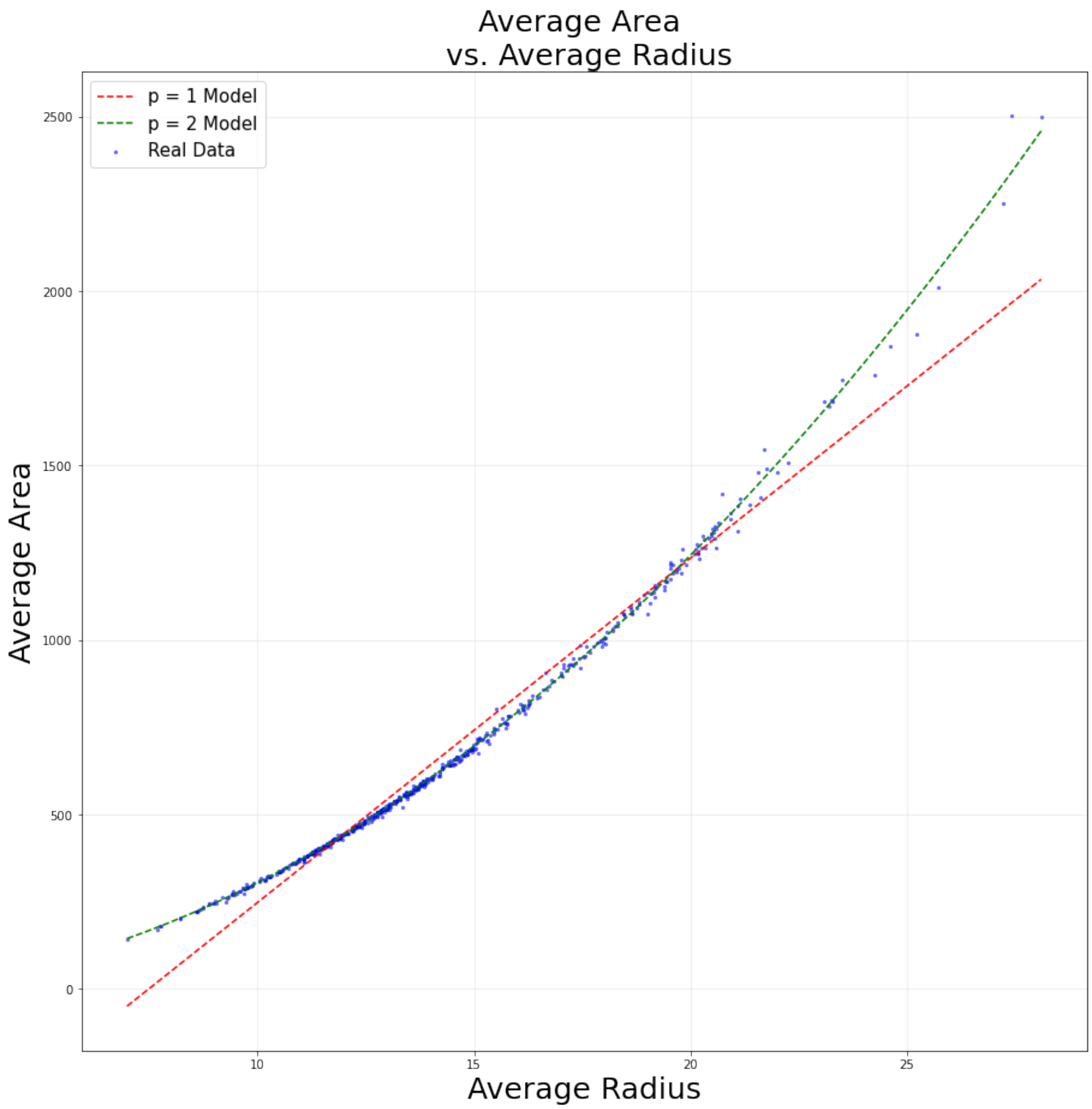
```
The Quadratic (p = 2) model coefficients are: [  3.10992516   0.43684601
-10.5164038 ]
The Quadratic (p = 2) model residuals are:
101     -0.593434
539     -6.400058
538      0.161050
568      0.854249
```

```
46       -0.071393
           ...
82     -100.563778
352     -49.596420
180     -55.606421
461     161.320947
212      39.860564
Name: res, Length: 569, dtype: float64
```

5. Plotting Data vs. Polynomial Models

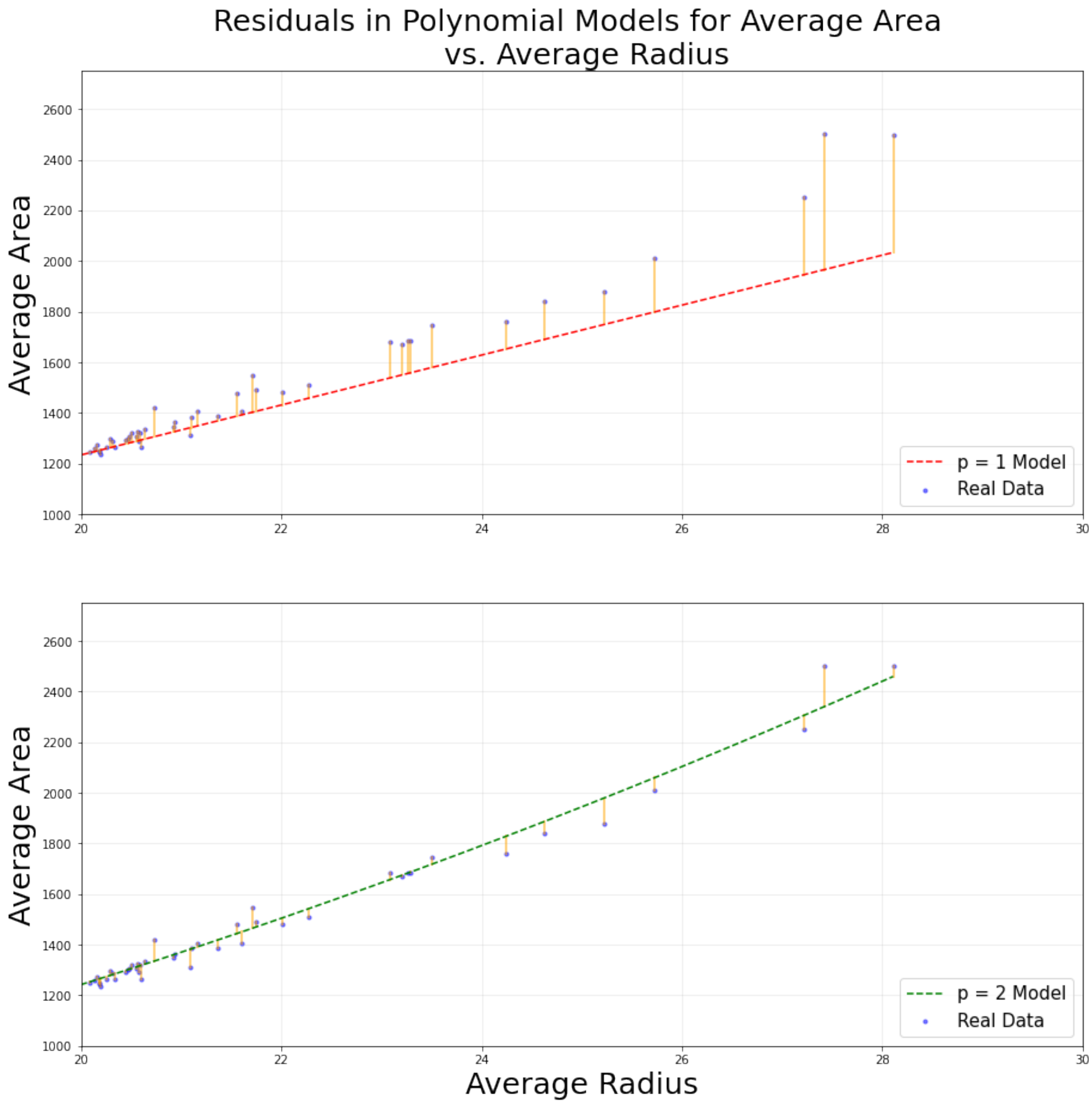
[16]:
```python
p1_coeffs = ols_all_models_dict['p = 1']['beta_flip']
p2_coeffs = ols_all_models_dict['p = 2']['beta_flip']

p1_data_df = ols_all_models_dict['p = 1']['model_df']
p2_data_df = ols_all_models_dict['p = 2']['model_df']

x_r = p1_data_df['radius_mean']
y_r = p1_data_df['area_mean']
p1_y_m = p1_data_df['y_m']
p2_y_m = p2_data_df['y_m']

x_m = np.linspace(min(x_r), max(x_r))
p1_y_m_linspace = np.polyval(p1_coeffs, x_m)
p2_y_m_linspace = np.polyval(p2_coeffs, x_m)
```

[17]:
```python
plt.figure(figsize = (15, 15))

plt.scatter(x_r, y_r, s=5, c='b', alpha=0.5, label = 'Real Data')
plt.plot(x_m, p1_y_m_linspace, 'r--', label = 'p = 1 Model')
plt.plot(x_m, p2_y_m_linspace, 'g--', label = 'p = 2 Model')
plt.grid(alpha = .25)

plt.title('Average Area \n vs. Average Radius', fontsize = 25)
plt.legend(fontsize = 15)
plt.xlabel('Average Radius', fontsize = 25)
plt.ylabel('Average Area', fontsize = 25)
plt.show()
```



[18]:
```
plt.figure(figsize = (15, 15))

plt.subplot(2, 1, 1)

for i in range(len(x_r)):
        p1_res_x = (x_r[i], x_r[i])
        p1_res_y = (y_r[i], p1_y_m[i])
        plt.plot(p1_res_x, p1_res_y, color = 'orange', linewidth = 2, alpha = 0.
 ↪5)

plt.scatter(x_r, y_r, s=10, c='b', alpha=0.5, label = 'Real Data')
plt.plot(x_m, p1_y_m_linspace, 'r--', label = 'p = 1 Model')
```

```
plt.grid(alpha = .25)
plt.title('Residuals in Polynomial Models for Average Area \n vs. Average␣
↪Radius', fontsize = 25)
plt.legend(fontsize = 15, loc = 'lower right')
plt.ylabel('Average Area', fontsize = 25)
plt.xlim([20, 30])
plt.ylim([1000, 2750])

plt.subplot(2, 1, 2)

for i in range(len(x_r)):
        p2_res_x = (x_r[i], x_r[i])
        p2_res_y = (y_r[i], p2_y_m[i])
        plt.plot(p2_res_x, p2_res_y, color = 'orange', linewidth = 2, alpha = 0.
↪5)

plt.scatter(x_r, y_r, s=10, c='b', alpha=0.5, label = 'Real Data')
plt.plot(x_m, p2_y_m_linspace, 'g--', label = 'p = 2 Model')

plt.grid(alpha = .25)
plt.legend(fontsize = 15, loc = 'lower right')
plt.xlabel('Average Radius', fontsize = 25)
plt.ylabel('Average Area', fontsize = 25)
plt.xlim([20, 30])
plt.ylim([1000, 2750])

plt.show()
```


Residuals in Polynomial Models for Average Area
vs. Average Radius
Average Area
2600
2400
2200
2000
1800
1600
1400
1200
1000
20
22
24
26
28
30
p = 1 Model
Real Data
p = 2 Model
Real Data
Average Radius