CS109A Introduction to Data Science:

Homework 2: Linear and k-NN Regression

Harvard University
Fall 2018
Instructors: Pavlos Protopapas, Kevin Rader

In [220]:
```
#RUN THIS CELL
import requests
from IPython.core.display import HTML
styles = requests.get("https://raw.githubusercontent.com/Harvard-IACS/2018-CS
HTML(styles)
```

Out[220]:

INSTRUCTIONS

- To submit your assignment follow the instructions given in canvas.
- Restart the kernel and run the whole notebook again before you submit.
- If you submit individually and you have worked with someone, please include the name of your [one] partner below.
- As much as possible, try and stick to the hints and functions we import at the top of the homework, as those are the ideas and tools the class supports and is aiming to teach. And if a problem specifies a particular library you're required to use that library, and possibly others from the import list.

Names of people you have worked with goes here: Xi Han, Haoran Zhao

---

In [283]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from statsmodels.api import OLS
%matplotlib inline
```

Predicting Taxi Pickups in NYC

In this homework, we will explore k-nearest neighbor and linear regression methods for predicting a quantitative variable. Specifically, we will build regression models that can predict the number of taxi pickups in New York city at any given time of the day. These prediction models will be useful, for example, in monitoring traffic in the city.

The data set for this problem is given in the file `dataset_1.csv`. You will need to separate it into training and test sets. The first column contains the time of a day in minutes, and the second column contains the number of pickups observed at that time. The data set covers taxi pickups recorded in NYC during Jan 2015.

We will fit regression models that use the time of the day (in minutes) as a predictor and predict the average number of taxi pickups at that time. The models will be fitted to the training set and evaluated on the test set. The performance of the models will be evaluated using the R^2 metric.

Question 1 [25 pts]

1.1. Use pandas to load the dataset from the csv file `dataset_1.csv` into a pandas data frame. Use the `train_test_split` method from `sklearn` with a `random_state` of 42 and a `test_size` of 0.2 to split the dataset into training and test sets. Store your train set dataframe in the variable `train_data`. Store your test set dataframe in the variable `test_data`.

1.2. Generate a scatter plot of the training data points with clear labels on the x and y axes. The time of the day on the x-axis and the number of taxi pickups on the y-axis. Make sure to title your plot.

1.3. Does the pattern of taxi pickups make intuitive sense to you?

Answers

1.1 Use pandas to load the dataset from the csv file ...

In [222]:
```
# read the file
# your code here
data = pd.read_csv("data/dataset_1.csv")
```

In [223]:
```
# split the data
# your code here
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42
```

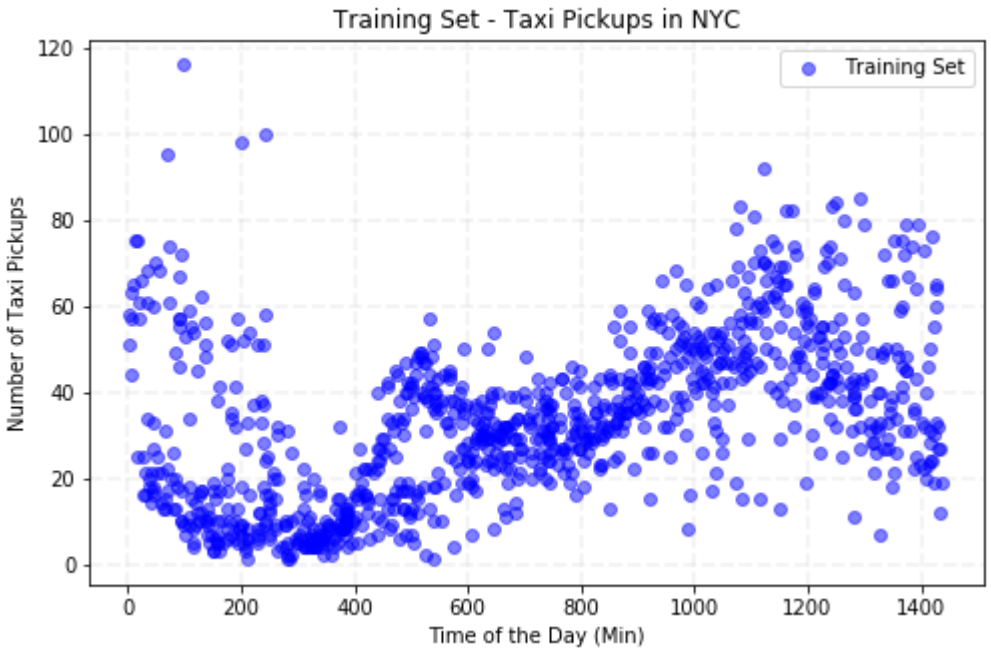
In [224]:
```
# Test size is indeed 20% of total
# your code here
print("Shape of full dataset is: {}".format(df.shape))
print("Shape of training dataset is: {}".format(train_data.shape))
print("Shape of test dataset is: {}".format(test_data.shape))
print("Test size / total size: {}".format(test_data.shape[0]/data.shape[0]))
```

```
Shape of full dataset is: (1250, 2)
Shape of training dataset is: (1000, 2)
Shape of test dataset is: (250, 2)
Test size / total size: 0.2
```

1.2 Generate a scatter plot of the training data points

In [225]:
```
# Your code here
fig, ax = plt.subplots(1, 1, figsize=(8, 5))

ax.grid(True, lw=1.75, ls='--', alpha=0.15)
ax.scatter(train_data['TimeMin'], train_data['PickupCount'], c='b', alpha=0.5
ax.set_title(r'Training Set - Taxi Pickups in NYC')
ax.set_xlabel(r'Time of the Day (Min)')
ax.set_ylabel(r'Number of Taxi Pickups')
ax.legend(loc='upper right')
```

Out[225]: <matplotlib.legend.Legend at 0x286d4b54e10>



1.3 Discuss your results. Does the pattern of taxi pickups make intuitive sense to you?

your answer here

Yes, it makes sense. The number of pickups is low during the early morning from 2am to 8am. The number of pickups increases dramatically from 8 am during the morning commute hours and decreased a little bit after 10am. However, the number of pickups trends up again from 4pm until midnight.

Question 2 [25 pts]

In lecture we've seen k-Nearest Neighbors (k-NN) Regression, a non-parametric regression technique. In the following problems please use built in functionality from `sklearn` to run k-NN Regression.

2.1. Choose `TimeMin` as your feature variable and `PickupCount` as your response variable. Create a dictionary of `KNeighborsRegressor` objects and call it `KNNModels`. Let the key for your `KNNmodels` dictionary be the value of k and the value be the corresponding `KNeighborsRegressor` object. For $k \in \{1, 10, 75, 250, 500, 750, 1000\}$, fit k-NN regressor models on the training set (`train_data`).

2.2. For each k on the training set, overlay a scatter plot of the actual values of `PickupCount` vs. `TimeMin` with a scatter plot of **predictions** for `PickupCount` vs `TimeMin`. Do the same for the test set. You should have one figure with 2 x 7 total subplots; for each k the figure should have two subplots, one subplot for the training set and one for the test set.

Hints:

1. Each subplot should use different color and/or markers to distinguish k-NN regression prediction values from the actual data values.
2. Each subplot must have appropriate axis labels, title, and legend.
3. The overall figure should have a title.

2.3. Report the R^2 score for the fitted models on both the training and test sets for each k (reporting the values in tabular form is encouraged).

2.4. Plot, in a single figure, the R^2 values from the model on the training and test set as a function of k.

Hints:

1. Again, the figure must have axis labels and a legend.
2. Differentiate R^2 visualization on the training and test set by color and/or marker.
3. Make sure the k values are sorted before making your plot.

2.5. Discuss the results:

1. If n is the number of observations in the training set, what can you say about a k-NN regression model that uses $k = n$?
2. What does an R^2 score of 0 mean?

3. What would a negative R^2 score mean? Are any of the calculated R^2 you observe negative?
4. Do the training and test R^2 plots exhibit different trends? Describe.
5. How does the value of k affect the fitted model and in particular the training and test R^2 values?
6. What is the best value of k and what are the corresponding training/test set R^2 values?

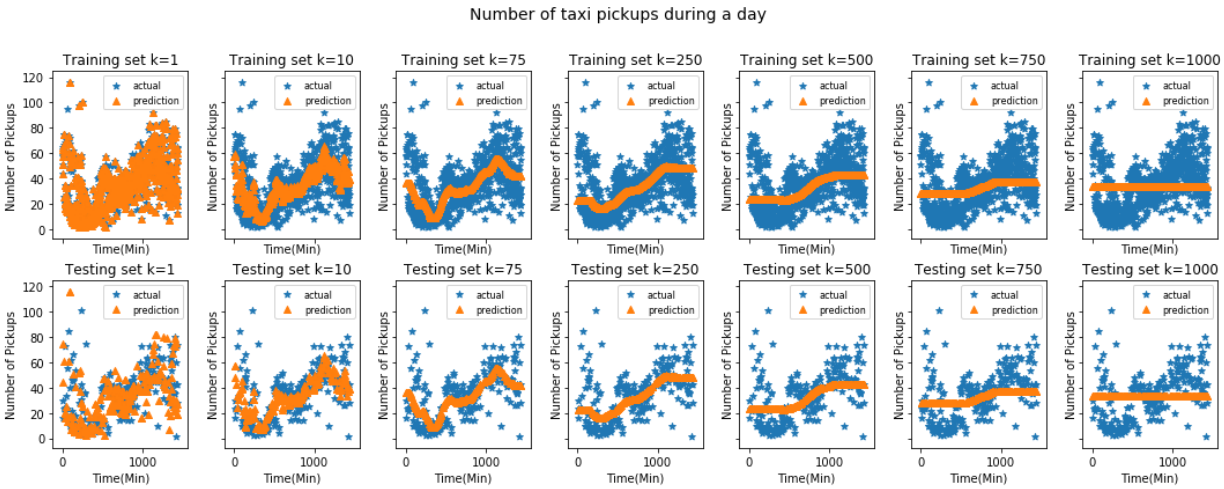
Answers

2.1 Choose `TimeMin` as your feature variable and `PickupCount` as your response variable. Create a dictionary ...

In [226]:
```
# your code here
KNNModels = {}
K = [1,10,75,250,500,750,1000]

for k in K:
    KNNModels[k] = KNeighborsRegressor(n_neighbors=k)
    KNNModels[k].fit(train_data[['TimeMin']], train_data[['PickupCount']])
```

2.2 For each k on the training set, overlay a scatter plot ...

In [230]:
```
f, axarr = plt.subplots(2, 7, figsize=(15, 6), sharex=True, sharey=True)

for i, k in enumerate(K):
    predicted_train = KNNModels[k].predict(train_data[['TimeMin']])
    axarr[0, i].scatter(train_data['TimeMin'], train_data['PickupCount'],mark
    axarr[0, i].scatter(train_data['TimeMin'], predicted_train,marker='^',lab
    axarr[0, i].set_title('Training set k=%i'%k)
    axarr[0, i].set_xlabel('Time(Min)')
    axarr[0, i].set_ylabel('Number of Pickups')
    axarr[0, i].legend(prop={'size': 8})

    predicted_test = KNNModels[k].predict(test_data[['TimeMin']])
    axarr[1, i].scatter(test_data['TimeMin'], test_data['PickupCount'],marker
    axarr[1, i].scatter(test_data['TimeMin'], predicted_test,marker='^',label
    axarr[1, i].set_title('Testing set k=%i'%k)
    axarr[1, i].set_ylabel('Number of Pickups')
    axarr[1, i].set_xlabel('Time(Min)')
    axarr[1, i].legend(prop={'size': 8})

f.suptitle('Number of taxi pickups during a day',fontsize=14)
f.tight_layout()
f.subplots_adjust(top=0.85)
```



2.3 Report the R^2 score for the fitted models ...

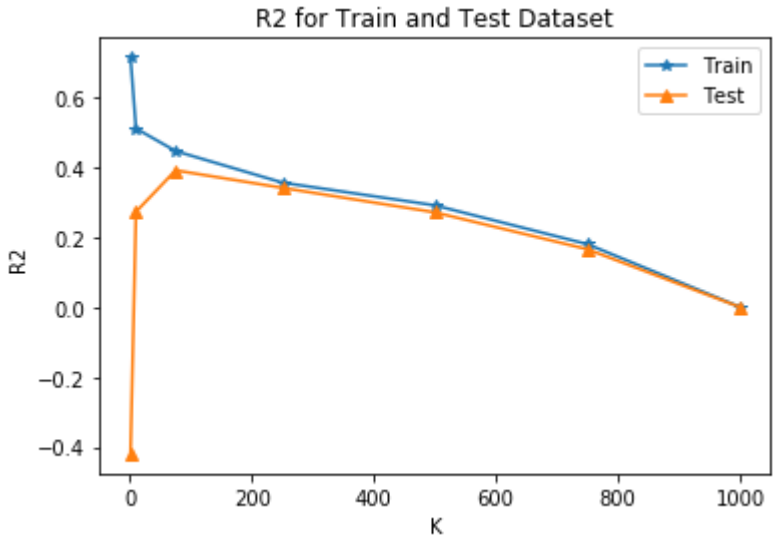
In [231]:
```
# your code here
r2_train = []
r2_test = []
K = [1,10,75,250,500,750,1000]

for k in K:
    r2_train.append(KNNModels[k].score(train_data[['TimeMin']],train_data[['P
    r2_test.append(KNNModels[k].score(test_data[['TimeMin']],test_data[['Pick

r2 = pd.DataFrame(np.transpose([K,r2_train,r2_test]), columns=['K', 'R2_Train
r2
```

Out[231]:

	K	R2_Train	R2_Test
0	1.0	0.712336	-0.418932
1	10.0	0.509825	0.272068
2	75.0	0.445392	0.390310
3	250.0	0.355314	0.340341
4	500.0	0.290327	0.270321
5	750.0	0.179434	0.164909
6	1000.0	0.000000	-0.000384

2.4 Plot, in a single figure, the R^2 values from the model on the training and test set as a function of k

In [232]:
```
# your code here
plt.plot(r2['K'], r2['R2_Train'], '*-', label='Train')
plt.plot(r2['K'], r2['R2_Test'], '^-', label='Test')
plt.legend()
plt.title('R2 for Train and Test Dataset')
plt.xlabel('K')
plt.ylabel('R2')
plt.show()
```



2.5 Discuss the results

your answer here

2.5.1. If n is the number of observations in the training set, what can you say about a k-NN regression model that uses $k = n$?

Answer: This means the predicted y value as a horizontal line which is the mean of all the training datasets

2.5.2. What does an R^2 score of 0 mean?

Answer: It means the model is performing the same as a horizontal line, the mean of all the observation

2.5.3. What would a negative R^2 score mean? Are any of the calculated R^2 you observe negative?

Answer: It means the model is even worse than a horizontal line, the mean of all the observations. Yes, k=1 and k=1000 have a negative R2 for test dataset

2.5.4. Do the training and test R^2 plots exhibit different trends? Describe.

Answer: Yes, when K is small, R2 on training datasets is very high but very low on the test dataset. As K increase to the point k=75, training R2 decreases but testing R2 decrease. They are geting closer and dcreasing together slowly after K=75

2.5.5. How does the value of k affect the fitted model and in particular the training and test R^2 values?

Answer: When K is small, R2 on training dataset are very high but very low on test dataset, which means the model is overfitted. R2 on training dataset is decreasing as K increasing. But R2 on test dataset increase initially as K increase until k=75. After that, R2 on test dataset also decreases as K increases.

2.5.6. What is the best value of k and what are the corresponding training/test set R^2 values?

Answer: K = 75 is the best value with Training R2=0.45 and Test R2 = 0.39 because it has the highest testing R2.

Question 3 [25 pts]

We next consider simple linear regression, which we know from lecture is a parametric approach for regression that assumes that the response variable has a linear relationship with the predictor. Use the `statsmodels` module for Linear Regression. This module has built-in functions to summarize the results of regression and to compute confidence intervals for estimated regression parameters.

3.1. Again choose `TimeMin` as your predictor and `PickupCount` as your response variable. Create a `OLS` class instance and use it to fit a Linear Regression model on the training set (`train_data`). Store your fitted model in the variable `OLSModel` .

3.2. Re-create your plot from 2.2 using the predictions from `OLSModel` on the training and test set. You should have one figure with two subplots, one subplot for the training set and one for the test set.

Hints:

1. Each subplot should use different color and/or markers to distinguish Linear Regression prediction values from that of the actual data values.
2. Each subplot must have appropriate axis labels, title, and legend.
3. The overall figure should have a title.

3.3. Report the R^2 score for the fitted model on both the training and test sets.

3.4. Report the slope and intercept values for the fitted linear model.

3.5. Report the 95% confidence interval for the slope and intercept.

3.6. Create a scatter plot of the residuals ($e = y - \hat{y}$) of the linear regression model on the training set as a function of the predictor variable (i.e. `TimeMin`). Place on your plot a horizontal line denoting the constant zero residual.

3.7. Discuss the results:

1. How does the test R^2 score compare with the best test R^2 value obtained with k-NN regression?
2. What does the sign of the slope of the fitted linear model convey about the data?
3. Based on the 95% confidence interval, do you consider the estimates of the model parameters to be reliable?
4. Do you expect 99% confidence intervals for the slope and intercept to be tighter or wider than the 95% confidence intervals? Briefly explain your answer.
5. Based on the residuals plot that you made, discuss whether or not the assumption of linearity is valid for this data.
6. Based on the data structure, what restriction on the model would you put at the endpoints (at $x \approx 0$ and $x \approx 1440$)? What does this say about the linearity assumption?

Answers

3.1 Again choose `TimeMin` as your predictor and `PickupCount` as your response variable. Create a `OLS` class instance ...

In [235]:
```
# your code here
x_train, y_train = train_data['TimeMin'], train_data['PickupCount']
x_test, y_test = test_data['TimeMin'], test_data['PickupCount']

# create the X matrix by appending a column of ones to x_train
X_train = sm.add_constant(x_train)

# build the OLS model from the training data
OLSModel = sm.OLS(y_train, X_train)

# fit and save regression info
texi_ols = OLSModel.fit()

print(texi_ols.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:            PickupCount   R-squared:                       0.24
3
Model:                            OLS   Adj. R-squared:                  0.24
2
Method:                 Least Squares   F-statistic:                     320.
4
Date:                Wed, 26 Sep 2018   Prob (F-statistic):           2.34e-6
2
Time:                        20:50:26   Log-Likelihood:                -4232.
9
No. Observations:                1000   AIC:                             847
0.
Df Residuals:                     998   BIC:                             848
0.
Df Model:                           1                                         
```

3.2 Re-create your plot from 2.2 using the predictions from `OLSModel` on the training and test set ...

In [236]:
```
# your code here
fig, ax = plt.subplots(1, 2, figsize=(16, 5))

ax[0].grid(True, lw=1.75, ls='--', alpha=0.15)
ax[0].scatter(x_train, y_train, c='b', alpha=0.5, label='Training Set')
ax[0].plot(x_train, texi_ols.fittedvalues, c='r', alpha=0.5, label='OLS Regre
ax[0].set_title(r'Training Set')
ax[0].set_xlabel(r'Time of the Day (Min)')
ax[0].set_ylabel(r'Number of Taxi Pickups')
ax[0].legend(loc='upper right')

ax[1].grid(True, lw=1.75, ls='--', alpha=0.15)
ax[1].scatter(x_test, y_test, c='g', alpha=0.5, label='Testing Set')
ax[1].plot(x_train, texi_ols.fittedvalues, c='r', alpha=0.5, label='OLS Regre
ax[1].set_title(r'Testing Set')
ax[1].set_xlabel(r'Time of the Day (Min)')
ax[1].set_ylabel(r'Number of Taxi Pickups')
ax[1].legend(loc='upper right')

fig.suptitle('Numer of taxi pickups during a day', fontsize=14)
```

Out[236]:
```
Text(0.5,0.98,'Numer of taxi pickups during a day')
```

3.3 Report the R^2 score for the fitted model on both the training and test sets.

In [240]:
```
# your code here
X_test = sm.add_constant(x_test)
print("R-squared for training set: %f" % texi_ols.rsquared)
print("R-squared for testing set: %f" % r2_score(y_test, texi_ols.predict(X_t
```

```
R-squared for training set: 0.243026
R-squared for testing set: 0.240662
```

3.4 Report the slope and intercept values for the fitted linear model.

In [241]:
```
# your code here
beta0 = texi_ols.params[0]
beta1 = texi_ols.params[1]
print("Intercept of the OLS: %f" % beta0)
print("Slope of the OLS: %f" % beta1)
```

```
Intercept of the OLS: 16.750601
Slope of the OLS: 0.023335
```

3.5 Report the 95% confidence interval for the slope and intercept.

In [242]:
```
# your code here
thresh = 0.05
intervals = texi_ols.conf_int(alpha=thresh)
intervals = intervals.rename(index=str, columns={0:str(thresh/2*100)+"%",1:st
intervals
```

Out[242]:

	2.5%	97.5%
const	14.675141	18.826062
TimeMin	0.020777	0.025893

3.6 Create a scatter plot of the residuals

In [260]:
```
# your code here
fig, ax = plt.subplots(1, 1, figsize=(8, 5))

ax.grid(True, lw=1.75, ls='--', alpha=0.15)
ax.scatter(x_train, texi_ols.resid, c='b', alpha=0.7, label='Residuals')
ax.plot(x_train, [0] * len(x_train), c='r', alpha=0.8, label='Constant Zero R
ax.set_title(r'Training Set - Regression residuals')
ax.set_xlabel(r'Time of the Day (Min)')
ax.set_ylabel(r'Residuals')
ax.legend(loc='upper right')
plt.show()
```

3.7 Discuss the results:

your answer here

1. How does the test R^2 score compare with the best test R^2 value obtained with k-NN regression?

Answer: R^2 for testing set in OLS is 0.240662, which is lower than the best test R^2 value obtained with k-NN regression of 0.390310.

2. What does the sign of the slope of the fitted linear model convey about the data?

Answer: Positive sign of the slope indicates that the number of texi pickups is generally positively correlated with the time in the day.

3. Based on the 95% confidence interval, do you consider the estimates of the model parameters to be reliable?

Answer: I would consider it reliable because the 95% confidence interval doesn't include 0.

4. Do you expect 99% confidence intervals for the slope and intercept to be tighter or wider than the 95% confidence intervals? Briefly explain your answer.

Answer: Wider. The wider the confidence interval is, the higher the chance for the parameters to fall within the range, the higher the confidence probability.

5. Based on the residuals plot that you made, discuss whether or not the assumption of linearity is valid for this data.

Answer: The assumption of linearity is not valid for this data, because residual terms are not independently, identically normally distributed, with very obvious clustering effect.

6. Based on the data structure, what restriction on the model would you put at the endpoints (at $x \approx 0$ and $x \approx 1440$)? What does this say about the linearity assumption?

Answer: x=0 and x=1400 are the same time of 12:00am. So I would put a constraint that the expected value of Y at point x = 0 equals the expected value of Y at point x = 1440. This violates the linearity assumption of indepence of residual terms, because the residal at x=0 would be the same residual at x=1440.

Outliers

You may recall from lectures that OLS Linear Regression can be susceptible to outliers in the data. We're going to look at a dataset that includes some outliers and get a sense for how that affects modeling data with Linear Regression. **Note, this is an open-ended question, there is not one correct solution (or one correct definition of an outlier).**

Question 4 [25 pts]

4.1. We've provided you with two files `outliers_train.txt` and `outliers_test.txt` corresponding to training set and test set data. What does a visual inspection of training set tell you about the existence of outliers in the data?

4.2. Choose `X` as your feature variable and `Y` as your response variable. Use `statsmodel` to create a Linear Regression model on the training set data. Store your model in the variable `OutlierOLSModel`.

4.3. You're given the knowledge ahead of time that there are 3 outliers in the training set data. The test set data doesn't have any outliers. You want to remove the 3 outliers in order to get the optimal intercept and slope. In the case that you're sure of the existence and number (3) of outliers ahead of time, one potential brute force method to outlier detection might be to find the best Linear Regression model on all possible subsets of the training set data with 3 points removed. Using this method, how many times will you have to calculate the Linear Regression coefficients on the training data?

4.4 In CS109 we're strong believers that creating heuristic models is a great way to build intuition. In that spirit, construct an approximate algorithm to find the 3 outlier candidates in the training data by taking advantage of the Linear Regression residuals. Place your algorithm in the function `find_outliers_simple`. It should take the parameters `dataset_x` and `dataset_y` representing your features and response variable values (make sure your response variable is stored as a numpy column vector). The return value should be a list `outlier_indices`

representing the indices of the 3 outliers in the original datasets you passed in. Remove the outliers that your algorithm identified, use `statsmodels` to create a Linear Regression model on the remaining training set data, and store your model in the variable `OutlierFreeSimpleModel`.

4.5 Create a figure with two subplots: the first is a scatterplot where the color of the points denotes the outliers from the non-outliers in the training set, and include two regression lines on this scatterplot: one fitted with the outliers included and one fitted with the outlier removed (all on the training set). The second plot should include a scatterplot of points from the test set with the same two regression lines fitted on the training set: with and without outliers. Visually which model fits the test set data more closely?

4.6. Calculate the R^2 score for the `OutlierOLSModel` and the `OutlierFreeSimpleModel` on the test set data. Which model produces a better R^2 score?

4.7. One potential problem with the brute force outlier detection approach in 4.3 and the heuristic algorithm you constructed 4.4 is that they assume prior knowledge of the number of outliers. In general you can't expect to know ahead of time the number of outliers in your dataset. Alter the algorithm you constructed in 4.4 to create a more general heuristic (i.e. one which doesn't presuppose the number of outliers) for finding outliers in your dataset. Store your algorithm in the function `find_outliers_general`. It should take the parameters `dataset_x` and `dataset_y` representing your features and response variable values (make sure your response variable is stored as a numpy column vector). It can take additional parameters as long as they have default values set. The return value should be the list `outlier_indices` representing the indices of the outliers in the original datasets you passed in (in the order of 'severity'). Remove the outliers that your algorithm identified, use `statsmodels` to create a Linear Regression model on the remaining training set data, and store your model in the variable `OutlierFreeGeneralModel`.

Hints:

1. How many outliers should you try to identify in each step?
2. If you plotted an R^2 score for each step the algorithm, what might that plot tell you about stopping conditions?

4.8. Run your algorithm in 4.7 on the training set data.

1. What outliers does it identify?
2. How do those outliers compare to the outliers you found in 4.4?
3. How does the general outlier-free Linear Regression model you created in 4.7 perform compared to the simple one in 4.4?

Answers

4.1 We've provided you with two files `outliers_train.txt` and `outliers_test.txt` corresponding to training set and test set data. What does a visual inspection of training set tell you about the existence of outliers in the data?

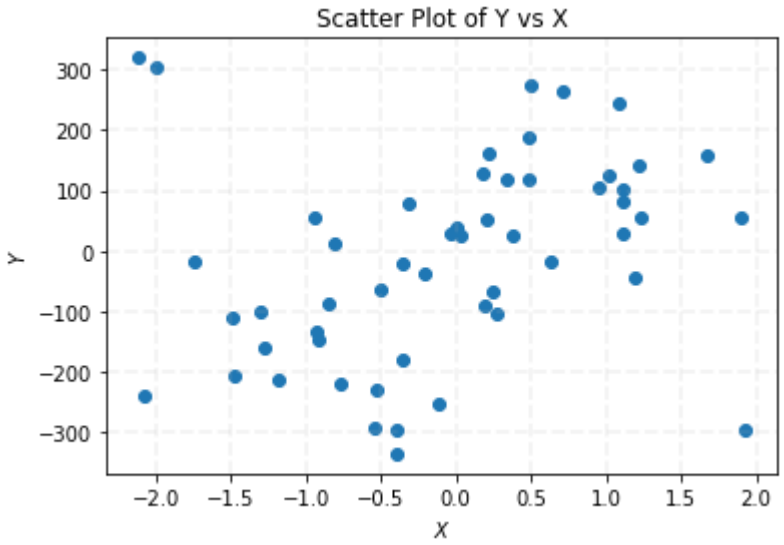
In [261]:
```
# read the data
# your code here
outliers_train = pd.read_csv("data/outliers_train.csv")
outliers_test = pd.read_csv("data/outliers_test.csv")
```

In [262]:
```
# your code here
def scatterplot(x, y, title):
    fig, ax = plt.subplots(1, 1)
    ax.grid(True, lw=1.75, ls='--', alpha=0.15)
    ax.scatter(x_train, y_train)
    ax.set_title(title)
    ax.set_xlabel(r'$X$')
    ax.set_ylabel(r'$Y$')

    return ax
```

In [263]:
```
# your code here
x_train, y_train = outliers_train['X'], outliers_train['Y']
x_test, y_test = outliers_test['X'], outliers_test['Y']
```

In [264]:
```
# scatter plot
# your code here
scatterplot(x_train, y_train, 'Scatter Plot of Y vs X')
```

Out[264]: <matplotlib.axes._subplots.AxesSubplot at 0x286d71b3940>



your answer here

Visual inspection of the training set tells me that some potential outliers are:

- Top Left Cornor: two points around (-2.0, 300)
- Bottom Right Cornor: one point around (2.0, -300)

4.2 Choose `X` as your feature variable and `Y` as your response variable. Use `statsmodel` to create ...

In [265]:
```
# your code here
# create the X matrix by appending a column of ones to x_train
X_train = sm.add_constant(x_train)

# build the OLS model from the training data
OutlierOLSModel = sm.OLS(y_train, X_train)

# fit and save regression info
results_sm = OutlierOLSModel.fit()

print(results_sm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.084
Model:                            OLS   Adj. R-squared:                  0.066
Method:                 Least Squares   F-statistic:                     4.689
Date:                Wed, 26 Sep 2018   Prob (F-statistic):             0.0351
Time:                        21:38:03   Log-Likelihood:                -343.59
No. Observations:                  53   AIC:                             691.2
Df Residuals:                      51   BIC:                             695.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -9.5063     22.192     -0.428      0.670     -54.059      35.046
X             47.3554     21.869      2.165      0.035       3.452      91.259
==============================================================================
Omnibus:                        2.102   Durbin-Watson:                   1.758
Prob(Omnibus):                  0.350   Jarque-Bera (JB):                1.251
Skew:                           0.215   Prob(JB):                        0.535
Kurtosis:                       3.617   Cond. No.                         1.06
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

4.3 One potential brute force method to outlier detection might be to find the best Linear Regression model on all possible subsets of the training set data with 3 points removed. Using this method, how many times will you have to calculate the Linear Regression coefficients on the training data?

your answer here

Picking 3 out of 53 samples gives in total (53 x 52 x 51) / (3 x 2) = 23,426 combinations.

4.4 CS109 hack ...

In [266]:
```
def find_outliers_simple(dataset_x, dataset_y):
    # your code here
    # create the X matrix by appending a column of ones to x_train
    X_train = sm.add_constant(dataset_x)

    # build the OLS model from the training data
    OutlierOLSModel = sm.OLS(dataset_y, X_train)

    # fit and save regression info
    results_sm = OutlierOLSModel.fit()

    # create a dataframe for labeling outliers
    df = pd.DataFrame({'X': dataset_x,
                       'Y': dataset_y,
                       'resid': results_sm.resid,
                       'resid_abs': abs(results_sm.resid)
                      })

    df.sort_values(by=['resid_abs'], ascending=False, inplace=True)

    outlier_indices = list(df.iloc[:3].index)

    return outlier_indices
```

In [267]:
```
# get outliers
# your code here
outlier_indices = find_outliers_simple(x_train, y_train)
print(outlier_indices)
```

```
[50, 51, 52]
```

In [268]:
```
outliers_train_clean = outliers_train.drop(outlier_indices)
```

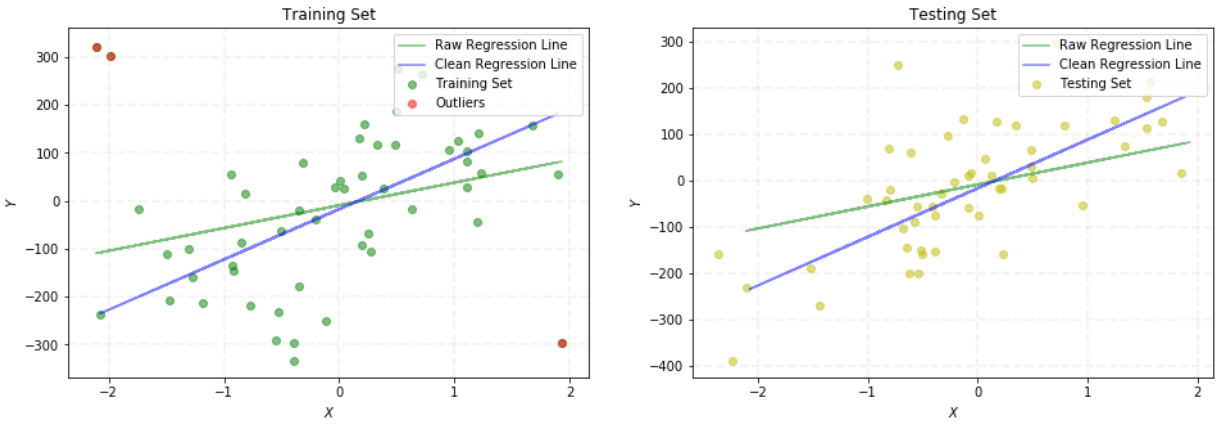
In [269]:
```
# calculate outlier model
# your code here
x_train_clean, y_train_clean = outliers_train_clean['X'], outliers_train_clea

# create the X matrix by appending a column of ones to x_train
X_train_clean = sm.add_constant(x_train_clean)

# build the OLS model from the training data
OutlierFreeSimpleModel = sm.OLS(y_train_clean, X_train_clean)

# fit and save regression info
results_sm_clean = OutlierFreeSimpleModel.fit()

print(results_sm_clean.summary())

```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.40
4
Model:                            OLS   Adj. R-squared:                  0.39
1
Method:                 Least Squares   F-statistic:                     32.5
0
Date:                Wed, 26 Sep 2018   Prob (F-statistic):           7.16e-0
7
Time:                        21:38:15   Log-Likelihood:                -309.2
1
No. Observations:                  50   AIC:                             622.
4
Df Residuals:                      48   BIC:                             626.
2
Df Model:                           1
```

4.5 Create a figure with two subplots: the first is a scatterplot ...

In [271]:
```
# plot
# your code here
outliers_x = outliers_train.loc[outlier_indices]['X']
outliers_y = outliers_train.loc[outlier_indices]['Y']

fig, ax = plt.subplots(1, 2, figsize=(16, 5))

ax[0].grid(True, lw=1.75, ls='--', alpha=0.15)
ax[0].scatter(x_train, y_train, c='g', alpha=0.5, label='Training Set')
ax[0].scatter(outliers_x, outliers_y, c='r', alpha=0.5, label='Outliers')
ax[0].plot(x_train, results_sm.fittedvalues, c='g', alpha=0.5, label='Raw Reg
ax[0].plot(x_train_clean, results_sm_clean.fittedvalues, c='b', alpha=0.5, la
ax[0].set_title(r'Training Set')
ax[0].set_xlabel(r'$X$')
ax[0].set_ylabel(r'$Y$')
ax[0].legend(loc='upper right')

ax[1].grid(True, lw=1.75, ls='--', alpha=0.15)
ax[1].scatter(x_test, y_test, c='y', alpha=0.5, label='Testing Set')
ax[1].plot(x_train, results_sm.fittedvalues, c='g', alpha=0.5, label='Raw Reg
ax[1].plot(x_train_clean, results_sm_clean.fittedvalues, c='b', alpha=0.5, la
ax[1].set_title(r'Testing Set')
ax[1].set_xlabel(r'$X$')
ax[1].set_ylabel(r'$Y$')
ax[1].legend(loc='upper right')
```

Out[271]: <matplotlib.legend.Legend at 0x286d8e6af60>



your answer here

Visually the regression line without outliers fit the test set data more closely.

4.6 Calculate the R^2 score for the `OutlierOLSModel` and the `OutlierFreeSimpleModel` on the test set data. Which model produces a better R^2 score?

In [272]:
```
# your code here
X_test = sm.add_constant(x_test)
r2_raw = r2_score(y_test, results_sm.predict(X_test))
r2_clean = r2_score(y_test, results_sm_clean.predict(X_test))

print("OutlierOLSModel R-squared: %f" % r2_raw)
print("OutlierFreeSimpleModel R-squared: %f" % r2_clean)
```

```
OutlierOLSModel R-squared: 0.340857
OutlierFreeSimpleModel R-squared: 0.452957
```

Conclusion: OutlierFreeSimpleModel is better than OutlierOLSModel.

4.7 One potential problem with the brute force outlier detection approach in 4.3 and the heuristic algorithm you constructed 4.4 is that they assume prior knowledge of the number of outliers.

In [276]:
```
# your code here
# Logic: Drop outliers until R-squared starts to decrease.
def find_outliers_general(dataset_x, dataset_y, n_trails):
    dataset_x_raw = dataset_x
    dataset_y_raw = dataset_y

    r2_score = []
    outliers = []

    for _ in range(n_trails):
        X_train = sm.add_constant(dataset_x)
        OutlierOLSModel = sm.OLS(dataset_y, X_train)
        results_sm = OutlierOLSModel.fit()

        # if r-squared starts to decrease, exit the function
        if r2_score and results_sm.rsquared < r2_score[-1]:
            break

        r2_score.append(results_sm.rsquared)

        df = pd.DataFrame({'X': dataset_x,
                           'Y': dataset_y,
                           'resid': results_sm.resid,
                           'resid_abs': abs(results_sm.resid)
                          })

        df.sort_values(by=['resid_abs'], ascending=False, inplace=True)

        outlier_indices = list(df.iloc[:1].index)
        outliers += outlier_indices
        df.drop(outlier_indices, inplace=True)
        dataset_x, dataset_y = df['X'], df['Y']

    return r2_score, outliers
```

4.8 Run your algorithm in 4.7 on the training set data

In [277]:
```
# your code here
r2_score, outliers = find_outliers_general(x_train, y_train, x_train.shape[0]
outliers_train.loc[outliers]
```

Out[277]:

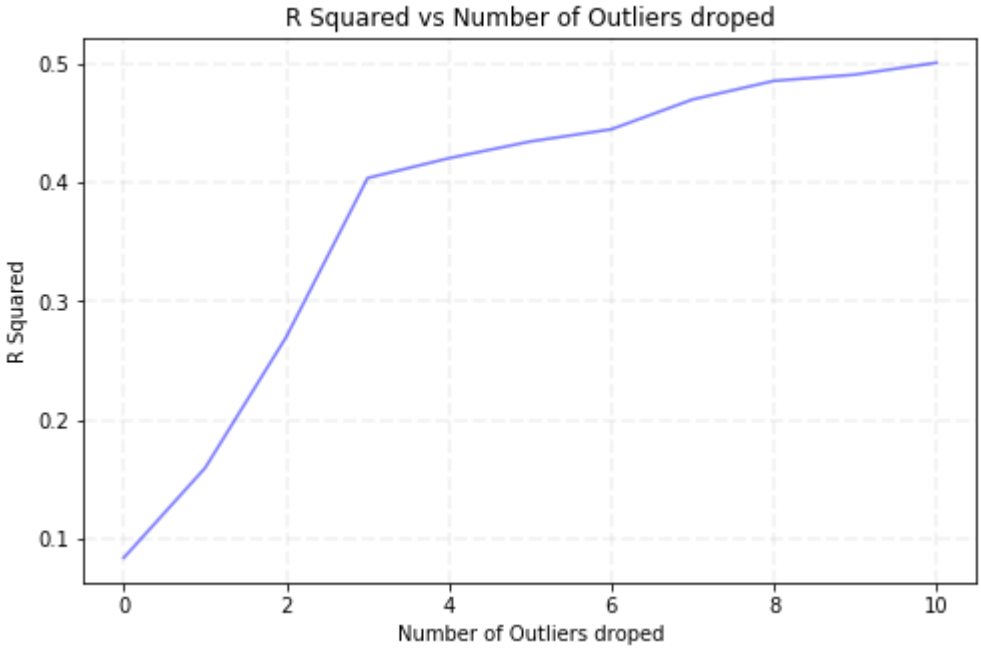
	X	Y
50	-2.110000	320.000000
51	-1.991000	303.000000
52	1.931000	-297.000000
1	-0.394034	-334.859357
14	-0.391668	-295.878637
28	0.502911	275.862982
5	-0.119129	-250.992560
35	-0.546692	-291.094951
24	0.716968	263.449637
20	-0.531202	-230.593281
7	1.085502	243.835916

In [278]:
```
# your code here
print(outliers)
```

```
[50, 51, 52, 1, 14, 28, 5, 35, 24, 20, 7]
```

In [279]:
```
# your code here
fig, ax = plt.subplots(1, 1, figsize=(8, 5))

ax.grid(True, lw=1.75, ls='--', alpha=0.15)
ax.plot(r2_score, c='b', alpha=0.5)
ax.set_title(r'R Squared vs Number of Outliers droped')
ax.set_xlabel(r'Number of Outliers droped')
ax.set_ylabel(r'R Squared')
```

Out[279]: Text(0,0.5,'R Squared')

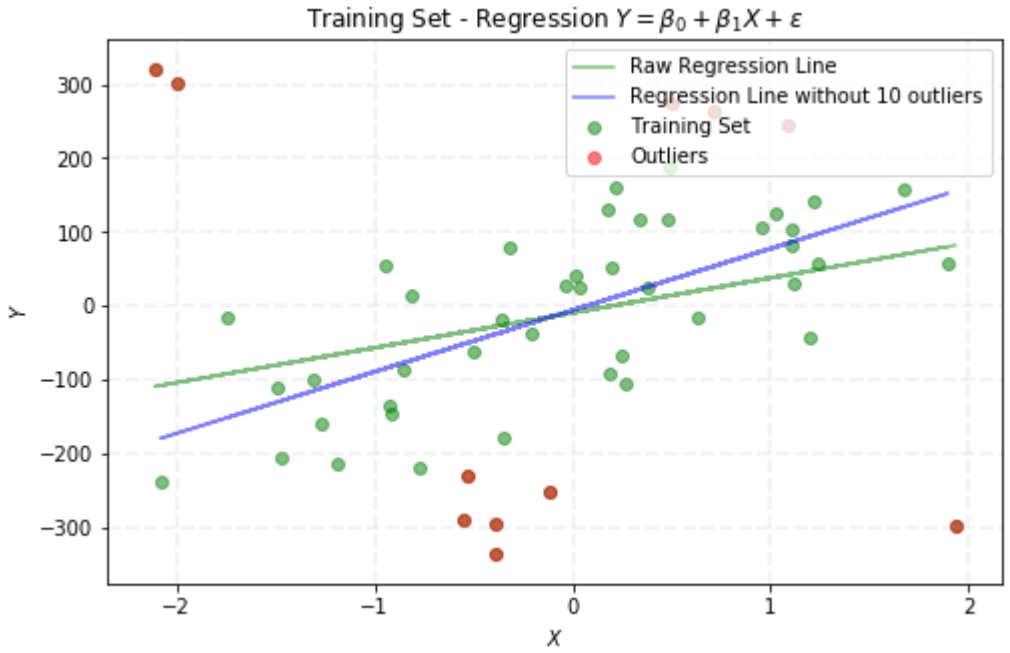


In [280]:
```
# your code here
outliers_x_gen = outliers_train.loc[outliers]['X']
outliers_y_gen = outliers_train.loc[outliers]['Y']

outliers_train_gen = outliers_train.drop(outliers)

x_train_gen, y_train_gen = outliers_train_gen['X'], outliers_train_gen['Y']

# create the X matrix by appending a column of ones to x_train
X = sm.add_constant(x_train_gen)

# build the OLS model from the training data
OutlierFreeGeneralModel = sm.OLS(y_train_gen, X)

# fit and save regression info
results_sm_gen = OutlierFreeGeneralModel.fit()

print(results_sm_gen.summary())

```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.48
8
Model:                            OLS   Adj. R-squared:                  0.47
5
Method:                 Least Squares   F-statistic:                     38.1
4
Date:                Wed, 26 Sep 2018   Prob (F-statistic):           2.68e-0
7
Time:                        21:40:27   Log-Likelihood:                -245.2
0
No. Observations:                  42   AIC:                             494.
4
Df Residuals:                      40   BIC:                             497.
9
Df Model:                           1
```

In [281]:
```
fig, ax = plt.subplots(1, 1, figsize=(8, 5))

ax.grid(True, lw=1.75, ls='--', alpha=0.15)
ax.scatter(x_train, y_train, c='g', alpha=0.5, label='Training Set')
ax.scatter(outliers_x_gen, outliers_y_gen, c='r', alpha=0.5, label='Outliers'
ax.plot(x_train, results_sm.fittedvalues, c='g', alpha=0.5, label='Raw Regres
ax.plot(x_train_gen, results_sm_gen.fittedvalues, c='b', alpha=0.5, label='Re
ax.set_title(r'Training Set - Regression $Y = \beta_0 + \beta_1 X + \epsilon$
ax.set_xlabel(r'$X$')
ax.set_ylabel(r'$Y$')
ax.legend(loc='upper right')
```

Out[281]: <matplotlib.legend.Legend at 0x286d8f69208>



In [284]:
```
# your code here
X_test = sm.add_constant(x_test)
r2_clean = r2_score(y_test, results_sm_clean.predict(X_test))
r2_gen = r2_score(y_test, results_sm_gen.predict(X_test))

print("OutlierFreeSimpleModel R-squared: %f" % r2_clean)
print("OutlierFreeGeneralModel R-squared: %f" % r2_gen)
```

```
OutlierFreeSimpleModel R-squared: 0.452957
OutlierFreeGeneralModel R-squared: 0.453556
```

your answer here

4.8.1 What outliers does it identify?

Answer: The following indices are identified as outliers: [50, 51, 52, 1, 14, 28, 5, 35, 24, 20, 7]

4.8.2 How do those outliers compare to the outliers you found in 4.4?

Answer: There are 10 outliers detected in the general method, because droping the 11th outliers will cause the r-squared to decrease, while 4.4 assumes that there are 3 outliers in total with the above scatter plot showing the distribution.

4.8.3 How does the general outlier-free Linear Regression model you created in 4.7 perform compared to the simple one in 4.4?

Answer: General model in 4.7 is slightly better than the model in 4.4, with R-squared of 0.453556 vs 0.452957.