a.

kind of learning: unsupervised

kind of task: clustering

short description (1-2 sentences) :
feature: eg. We could use word frequency to filter top n words in each article, and cluster those articles into different clusters using these top words and their frequency.
label: no labels since it's unsupervised.

kind of learning: supervised

kind of task: regression

short description (1-2 sentences) :
feature: default history, age, gender, income

Use historical data of customers to train model so that when features of a new person are input into the model, it can predict its credit situation and decide reasonable limit.

label: low risk, medium risk, high risk (with according credit allowance)

c

kind of learning: supervised

kind of task: classification

short description (1-2 sentences) :
feature: homework grade, mid-term grade, attendance
label: pass fail
Using the historical academic data of students to train the model, with above features and labels, so that when the feature of a new student is input into the model, it can predict whether the student can pass or fail.

d.

kind of learning: unsupervised

kind of task:

unsupervised task (clustering, dimensionality reduction)
short description (1-2 sentences) :
feature: purchase frequency, type of goods purchased, purchase amount
Use above features to fit the model and classify the customers into different clusters

e. **kind of learning:** supervised

kind of task: classification

short description (1-2 sentences) :
features: number of non-word characters. frequency of words appear frequently in spam e-mails, such as "gamble". E-mail domain name
label: spam, normal

2.a.

Histogram	binning
one-hot encoding	categorical feature
radar plot	> 2d data
box-cox transform	feature skewness
min-max	values in [0, 1]
robust z-score	outliers

2b.

	frequency distribution	median	mean
Nominal	X		
Ordinal		X	
Numerical			X

2C.

Yale:1000

School	Yale	Stanford	CMU	MIT	UCB
OHE	1000	0100	0010	0001	0000
Major	Engineering	Policy			
OHE	1	0			

Student	OHE
Yale Policy	[1,0,0,0,0]
Stanford Policy	[0,1,0,0,0]
Stanford Engineering	[0,1,0,0,1]
CMU Policy	[0,0,1,0,0]
MIT Engineering	[0,0,0,1,1]
CMU Engineering	[0,0,1,0,1]

new data matrix have 5 features/columns.

3a.(i) D

(ii)

$$\frac{\partial L}{\partial w0} = \sum(-2 * (y_i - (w_1 * x_i + w_0))) = 0$$

$$\sum(y_i - (w_1 * x_i + w_0)) = 0$$

$$\sum y_i \ = \sum(w_1 * x_i + w_0) = w1 * \sum x_i + n * w_0$$

$$\frac{\partial L}{\partial w1} = \sum(2 * \left(y_i - (w_1 * x_i + w_0)\right) * (-x_i)) = 0$$

$$\sum y_i x_i \ = \sum(w_1 x_i{}^2 + w_0 x_i) = w1 * \sum x_i{}^2 + w_0 * \ \sum x_i$$

Use this two function, we could get:

W0 = $\bar{y} - w_1\bar{x}$

W1=$\frac{n*\sum yixi-\sum yi*\sum xi}{n*\sum xi^2-(\sum xi)^2}$

b.(i) No. It exist overfitting or multicollinearity if we directly employ linear regression and use least square to minimize loss . Since there are too many features compare to the number of instances. The model can be complex to fit the train data but have bad performance on evaluation/test data. Also, those features can exist multicollinearity.

(ii)Use Lasso regularization to reduce model parameters. That is, when compute loss, we could add a lambda*R(w) (R(w) is a function related with w, w are weights for different features/arameters, lambda is hyperparameters to control the effect of R(w) to loss computation), R(w) can be Lasso regression.

The model can also apply VIF to decide if several features have multicollinearity. If so, delete some of the redundant features.

c. (iii)

d. Yes. You could use utility and utility^2 as two features/variables and apply linear regression model.

That is, Revenue=a*utility^2+b*utility+c

e. (i) complete multi-collinearity. Since these 3 features with different name are identical, variance of each of the 3 variables can be completely explained by other variables and thus have very large (infinite) VIF value. In this case, we can never know exactly which variables (if any) are truly predictive of response.

(ii) Compute VIF for each feature and drop those who have large VIF values.

f. Yes. Revenue=a*price^2+b*brand_perception+c*region^2+d*region+e

4. Conceptual

a.

	Bias	Variance
Linear regression	low / $\boxed{\text{high}}$	$\boxed{\text{low}}$ / high
Polynomial regression with degree 3	$\boxed{\text{low}}$ / high	$\boxed{\text{low}}$ / high
Polynomial regression with degree 10	$\boxed{\text{low}}$ / high	low / $\boxed{\text{high}}$

b.(i) For larger k, the approximation error will not change since the model complexity doesn't change, but the estimation error will become smaller since the model is unlikely to be overfit with larger dataset. Also, the computational tim will become higher since training dataset will become larger.

Approximation error	Estimation error	Computational time
( ) Smaller	(X) Smaller	( ) Lower
( ) Larger	( ) Larger	(X) Higher
(X) Stays the same	( ) Stays the same	( ) Stays the same

(ii) For smaller k, the approximation error will not change since the model complexity doesn't change, but the estimation error will become larger since the model tend to be overfit. Also, the computational tim will become lower since training dataset will become smaller.

Approximation error	Estimation error	Computational time
( ) Smaller	( ) Smaller	(X) Lower
( ) Larger	(X) Larger	( ) Higher
(X) Stays the same	( ) Stays the same	( ) Stays the same

c.(i) Overfitting. If the model is overfit, then it may fit the train data well with low bias, but cannot represent the real relationship among price and features, thus have good performance on training data and bad performance on testing data.

(ii)

(a) With a large dataset, we could use a small k to decrease computational time.

(b)With a small dataset, we could use a larger k or even k=n-1 to increase the training dataset to prevent overfitting.

1 Exploratory Data Analysis

Employee turnover is a key problem faced by many organizations. When good people leave, it usually costs the organization substantial time and other resources to find a replacement. Therefore, many organizations try to keep the churn rate at a low level. Imagine a company who now wants to understand its employee churn situation. Its HR (Human Resources) department gives you some data of their employees, and asks you to do exploratory data analysis and to predict employee churn.

You are free to choose any statistics library to analyze the data. In your answer, please include both the snippets of your code as well as the outputs.

Download the data `termination.csv` and `.ipynb` template from Canvas. Use the downloaded resources to answer the following questions:

1.1 a. (2 pts) Display a summary of the data (i.e. min, max, mean and quartiles for each variable). In the summary statistics, are there any meaningless quantities?

In [1]:
```
# Step 1: Load essential packages -- refer to recitation
import pandas as pd
import numpy as np
```

In [2]:
```
# Step 2: load data using read_csv function
```

In [2]:
```
df=pd.read_csv('termination.csv',sep=',')
```

In [3]:
```
# step 3: Invoke appropriate function on the loaded data to get the summary statist
```

In [4]:
```
print(df.describe(include='all'))
```

```
         EmployeeID   recorddate_key birthdate_key orighiredate_key  \
count  49653.000000            49653         49653            49653
unique          NaN              130          5342             4415
top             NaN  12/31/2013 0:00    1954-08-04       2004-12-04
freq            NaN             5215            40               50
mean    4859.495740              NaN           NaN              NaN
std     1826.571142              NaN           NaN              NaN
min     1318.000000              NaN           NaN              NaN
25%     3360.000000              NaN           NaN              NaN
50%     5031.000000              NaN           NaN              NaN
75%     6335.000000              NaN           NaN              NaN
max     8336.000000              NaN           NaN              NaN

       terminationdate_key           age  length_of_service  city_name  \
count                49653  49653.000000       49653.000000      49653
unique                1055           NaN                NaN         40
top             1900-01-01           NaN                NaN  Vancouver
freq                 42450           NaN                NaN      11211
mean                   NaN     42.077035          10.434596        NaN
std                    NaN     12.427257           6.325286        NaN
min                    NaN     19.000000           0.000000        NaN
25%                    NaN     31.000000           5.000000        NaN
50%                    NaN     42.000000          10.000000        NaN
75%                    NaN     53.000000          15.000000        NaN
max                    NaN     65.000000          26.000000        NaN

       department_name    job_title    store_name gender_short gender_full  \
count            49653        49653  49653.000000        49653       49653
unique              21           47           NaN            2           2
top              Meats  Meat Cutter           NaN            F      Female
freq             10269         9984           NaN        25898       25898
mean               NaN          NaN     27.297605          NaN         NaN
std                NaN          NaN     13.514134          NaN         NaN
min                NaN          NaN      1.000000          NaN         NaN
25%                NaN          NaN     16.000000          NaN         NaN
50%                NaN          NaN     28.000000          NaN         NaN
75%                NaN          NaN     42.000000          NaN         NaN
max                NaN          NaN     46.000000          NaN         NaN

       termreason_desc   termtype_desc   STATUS_YEAR  STATUS BUSINESS_UNIT
count            49653           49653  49653.000000   49653         49653
unique               4               3           NaN       2             2
top     Not Applicable  Not Applicable           NaN  ACTIVE        STORES
freq             41853           41853           NaN   48168         49068
mean               NaN             NaN   2010.612612     NaN           NaN
std                NaN             NaN      2.845577     NaN           NaN
min                NaN             NaN   2006.000000     NaN           NaN
25%                NaN             NaN   2008.000000     NaN           NaN
50%                NaN             NaN   2011.000000     NaN           NaN
75%                NaN             NaN   2013.000000     NaN           NaN
max                NaN             NaN   2015.000000     NaN           NaN
```

The statistical data(mean,std,etc.) of record_date_kyey, orighiredate_key.terminationdate_key, city_name,department_name, job_title, gender_short, gender_full, termreason_desc,termtype_desc,STATUS,BUSINESS_UNIT are meaningless.

1.2 b. (5 pts) The data include 10 years (2006 - 2015) of records for both active and terminated employees. Status Year field shows the year of data, and Status field shows the employment status – ACTIVE or TERMINATED in the corresponding status year. The company is interested in what proportion of the staff are leaving. Compute: 1) the percent of terminated employees out of all employees for each year; 2) average termination rate over the 10 years?

In [5]:
```
# Step 1: Create a pivot_table indexing STATUS_YEAR and apply to STATUS column
```

In [6]:
```
pv_table=df.pivot_table(values='EmployeeID',index='STATUS_YEAR',columns='STATUS',ag
pv_table
```

Out[6]:

STATUS	ACTIVE	TERMINATED
STATUS_YEAR		
2006	4445	134
2007	4521	162
2008	4603	164
2009	4710	142
2010	4840	123
2011	4972	110
2012	5101	130
2013	5215	105
2014	4962	253
2015	4799	162

In [7]:
```
# Step 2: Based on the pivot_table, find total number of employees each year
```

In [8]:
```
n_of_employee_year=pv_table.sum(axis=1)
n_of_employee_year
```

Out[8]:
```
STATUS_YEAR
2006    4579
2007    4683
2008    4767
2009    4852
2010    4963
2011    5082
2012    5231
2013    5320
2014    5215
2015    4961
dtype: int64
```

In [9]:
```
# Step 3: Now compute the percentage
```

In [10]:
```
pv_percent=pv_table['TERMINATED'].div(n_of_employee_year,axis=0)
pv_percent.apply(lambda x: format(x,'.4%'))
```

Out[10]:
```
STATUS_YEAR
2006    2.9264%
2007    3.4593%
2008    3.4403%
2009    2.9266%
2010    2.4783%
2011    2.1645%
2012    2.4852%
2013    1.9737%
2014    4.8514%
2015    3.2655%
dtype: object
```

In [11]:
```
# Step 4: Invoke a function to compute average on the calculated percentage.
```

In [17]:
```
print('{:.4%}'.format(pv_percent.mean()))
print('*Note: here I use mean of the termination rates of ten years rather than dire
```

```
2.9971%
*Note: here I use mean of the termination rates of ten years rather than directly use t
he termination number of ten years to compute
```

1.3 c.(5 pts) In addition to the proportion of terminated employees, the company wants to know more about different types of termination. Give a stacked bar chart of terminates, where x-axis is status year, y-axis is number of terminated employees, and different colors in a bar show different termination reasons ('termreason desc' field in the data). What do you observe in this plot?

In [ ]:
```
# Step 1: Filter dataframe for the relevant status for this question
```

In [118]:
```
filter_data=df[df['STATUS']=='TERMINATED'][['termreason_desc','STATUS_YEAR','STATUS
filter_data
```

Out[118]:

	termreason_desc	STATUS_YEAR	STATUS
48168	Retirement	2010	TERMINATED
48169	Retirement	2011	TERMINATED
48170	Retirement	2006	TERMINATED
48171	Retirement	2011	TERMINATED
48172	Retirement	2012	TERMINATED
48173	Resignaton	2011	TERMINATED
48174	Resignaton	2012	TERMINATED
48175	Resignaton	2015	TERMINATED
48176	Resignaton	2011	TERMINATED
48177	Resignaton	2011	TERMINATED
48178	Retirement	2006	TERMINATED
48179	Retirement	2009	TERMINATED

In [53]:
```
# Step 2: Similar to part (b) create pivot table on column termreason_desc
```

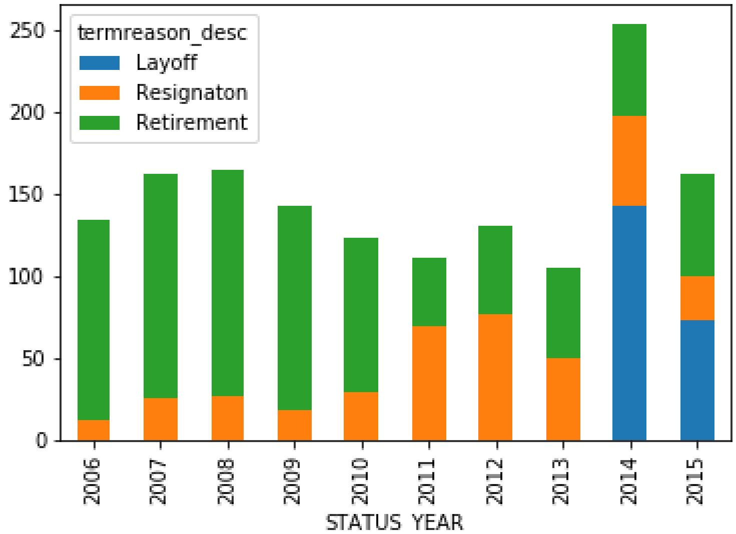
In [132]:
```
filtert_pv=filter_data.pivot_table('STATUS',columns='termreason_desc',index='STATUS
filtert_pv
```

Out[132]:

termreason_desc	Layoff	Resignaton	Retirement
STATUS_YEAR			
2006	NaN	12.0	122.0
2007	NaN	25.0	137.0
2008	NaN	26.0	138.0
2009	NaN	18.0	124.0
2010	NaN	29.0	94.0
2011	NaN	69.0	41.0
2012	NaN	76.0	54.0
2013	NaN	49.0	56.0
2014	142.0	55.0	56.0
2015	73.0	26.0	63.0

In [57]:
```
# Step 3: Plot stacked bar chart using pandas plot bar function
```

In [137]:
```
filtert_pv.plot.bar(stacked=True);
```



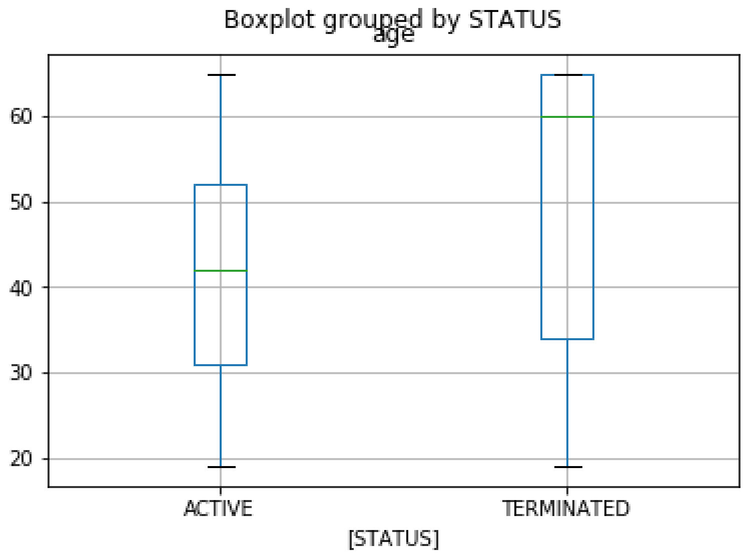
1.4 d. (3 pts) Does Age affect termination? Draw (2) Box-plots of Age for active and terminated employees separately. What does the box-plot tell you?

In [58]:
```
# Step 1: Use pandas boxplot for this part
```

In [147]:
```
data_box=df[['age','STATUS']]
data_box
```

49639	59	TERMINATED
49640	64	TERMINATED
49641	54	TERMINATED
49642	62	TERMINATED
49643	30	TERMINATED
49644	39	TERMINATED
49645	21	TERMINATED
49646	24	TERMINATED
49647	21	TERMINATED
49648	24	TERMINATED
49649	65	TERMINATED

In [146]:
```
data_box.boxplot(column=['age'],by=['STATUS']);
```



1 Linear Regression and Model Selection on Advertising Data

Let's take a look at some advertising data, and then fill in the missing code using the provided hints and answer the questions in 1-2 sentences (Each ... indicates missing code or answer).

What are the **features**?

- TV: advertising dollars spent on TV for a single product in a given market (in thousands of dollars)
- Radio: advertising dollars spent on Radio 1
- Radio 2: advertising dollars spent on Radio 2
- Newspaper: advertising dollars spent on Newspaper
- Area: the location

What is the **response**?

- Sales: sales of a single product in a given market (in thousands of widgets)

There are 200 **observations**, and thus 200 markets in the dataset.

In [1]:

```
# imports
import pandas as pd
import numpy as np

##fill in the missing code to read data into a DataFrame using read_csv in pandas p
data = pd.read_csv('Advertising.csv', index_col=0)
data = data.dropna()
data.head()
```

Out[1]:

	TV	radio	radio_2	newspaper	area	sales
1	230.1	37.8	75.6	69.2	rural	22.1
2	44.5	39.3	78.6	45.1	urban	10.4
3	17.2	45.9	91.8	69.3	rural	9.3
4	151.5	41.3	82.6	58.5	urban	18.5
5	180.8	10.8	21.6	58.4	suburban	12.9

1.1 Fitting the data to a Linear model (4 pts)

Let's try to fitting a linear regression model immediately to the given dataset

In [2]:

```
from sklearn.linear_model import LinearRegression

##fill in the feature list
feature_cols = ['TV','radio','radio_2','newspaper','area']
X = data[feature_cols].values
y = data.sales.values

## instantiate a Linear Regression model and fit to the data
model = LinearRegression()
model.fit(X,y)
# print coefficients
print(feature_cols, model.coef_)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-2-fb0329f47b70> in <module>
      8 ## instantiate a Linear Regression model and fit to the data
      9 model = LinearRegression()
---> 10 model.fit(X, y)
     11 # print coefficients
     12 print(feature_cols, model.coef_)

~\Anaconda3\lib\site-packages\sklearn\linear_model\base.py in fit(self, X, y, sample_weight)
    461         n_jobs_ = self.n_jobs
    462         X, y = check_X_y(X, y, accept_sparse=['csr', 'csc', 'coo'],
--> 463                          y_numeric=True, multi_output=True)
    464 
    465         if sample_weight is not None and np.atleast_1d(sample_weight).ndim > 1:

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, warn_on_dtype, estimator)
    717                     ensure_min_features=ensure_min_features,
    718                     warn_on_dtype=warn_on_dtype,
--> 719                     estimator=estimator)
    720     if multi_output:
    721         y = check_array(y, 'csr', force_all_finite=True, ensure_2d=False,

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, warn_on_dtype, estimator)
    494             try:
    495                 warnings.simplefilter('error', ComplexWarning)
--> 496                 array = np.asarray(array, dtype=dtype, order=order)
    497             except ComplexWarning:
    498                 raise ValueError("Complex data not supported\n"

~\Anaconda3\lib\site-packages\numpy\core\numeric.py in asarray(a, dtype, order)
    536 
    537     """
--> 538     return array(a, dtype, copy=False, order=order)
```

```
    539
    540

ValueError: could not convert string to float: 'rural'
```

Question: What is the output of this first attempt to fit a Linear Regression model? Explain the output in 1-2 sentences.

It reports an error, since the data of rural feature is not in numerical type. So it cannot be fit into the model

1.2 Handling Categorical Features via One-Hot Encoding (4 pts)

We have to represent **area** numerically, but we can't simply code it as 0=rural, 1=suburban, 2=urban because that would imply an **ordered relationship** between suburban and urban (and thus urban is somehow "twice" the suburban category).

Question: How many variables need to be created and why?

(Put your answer here) ...

Question: Intepret the encoding

- **rural** is coded as ...
- **suburban** is coded as ...
- **urban** is coded as ...

In [133]:

```python
## create three dummy variables using get_dummies, then exclude the first dummy col
area_dummies =pd.get_dummies(data['area'],drop_first=True)#get 3 columns

# concatenate the dummy variable columns onto the original DataFrame (axis=0 means
data = pd.concat([data, area_dummies], axis=1)
data.head()
```

Out[133]:

	TV	radio	radio_2	newspaper	area	sales	suburban	urban
1	230.1	37.8	75.6	69.2	rural	22.1	0	0
2	44.5	39.3	78.6	45.1	urban	10.4	0	1
3	17.2	45.9	91.8	69.3	rural	9.3	0	0
4	151.5	41.3	82.6	58.5	urban	18.5	0	1
5	180.8	10.8	21.6	58.4	suburban	12.9	1	0

Let's include the new dummy variables in the model:

In [134]:

```
## your new list of features
feature_cols = ['TV','radio','radio_2','newspaper','suburban','urban']
X = data[feature_cols].values

from sklearn import preprocessing

## Min max feature scaling using MinMaxScaler from sklearn
min_max_scaler = preprocessing.MinMaxScaler()
X = min_max_scaler.fit_transform(X)

# instantiate, fit
model = LinearRegression()
model.fit(X,y)

# print coefficients
print(feature_cols, model.coef_)
```

```
['TV', 'radio', 'radio_2', 'newspaper', 'suburban', 'urban'] [13.54373938  4.6528972
 4.6528972  -0.11576352 -0.1178903   0.25352196]
```

Question: Holding all other variables fixed, how do we interpret the coefficients of dummy variables?

- Being a **suburban** area is associated with an average -0.1158 less sales than in rural
- Being an **urban** area is associated with an average 0.25 more sales than in rural

Question: What are the coefficients of radio and radio_2 features are look like? How does it possibly happen (1-2 sentences)?

(Put your answer here) That 2 coefficient are almost the same. It might be collinearity exist between them.

1.3 Handling Collinearity using VIF (4 pts)

In [53]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

## use variance_inflation_factor to compute VIF scores for the features
vif = [variance_inflation_factor(X,i) for i in range(X.shape[1])]

print(vif)
```

```
[2.913123986087023, inf, inf, 3.1312420587779055, 1.6205239662516826, 1.747521783948640
6]
```

Question: Based on the VIF values, what features are collinear and what features can we remove to elimiate collinearity?

Feature 'radio' and 'radio_2' are collinear, we could remove 'radio_2'

Recompute VIF scores after removing that feature

In [137]:
```
## Your new list of features after removing collinear features
feature_cols = ['TV','radio','newspaper','suburban','urban']
X = data[feature_cols].values

## Min max feature scaling
min_max_scaler = preprocessing.MinMaxScaler()
X = min_max_scaler.fit_transform(X)

## Compute VIF scores for all remaining features
vif = [variance_inflation_factor(X,i) for i in range(X.shape[1])]

print(vif)
```

```
[2.8283216335767465, 3.519459817796587, 3.0846681025604923, 1.7721591778172374, 1.68876
02573497236]
```

Question: How do the VIF scores for these fetures look like?

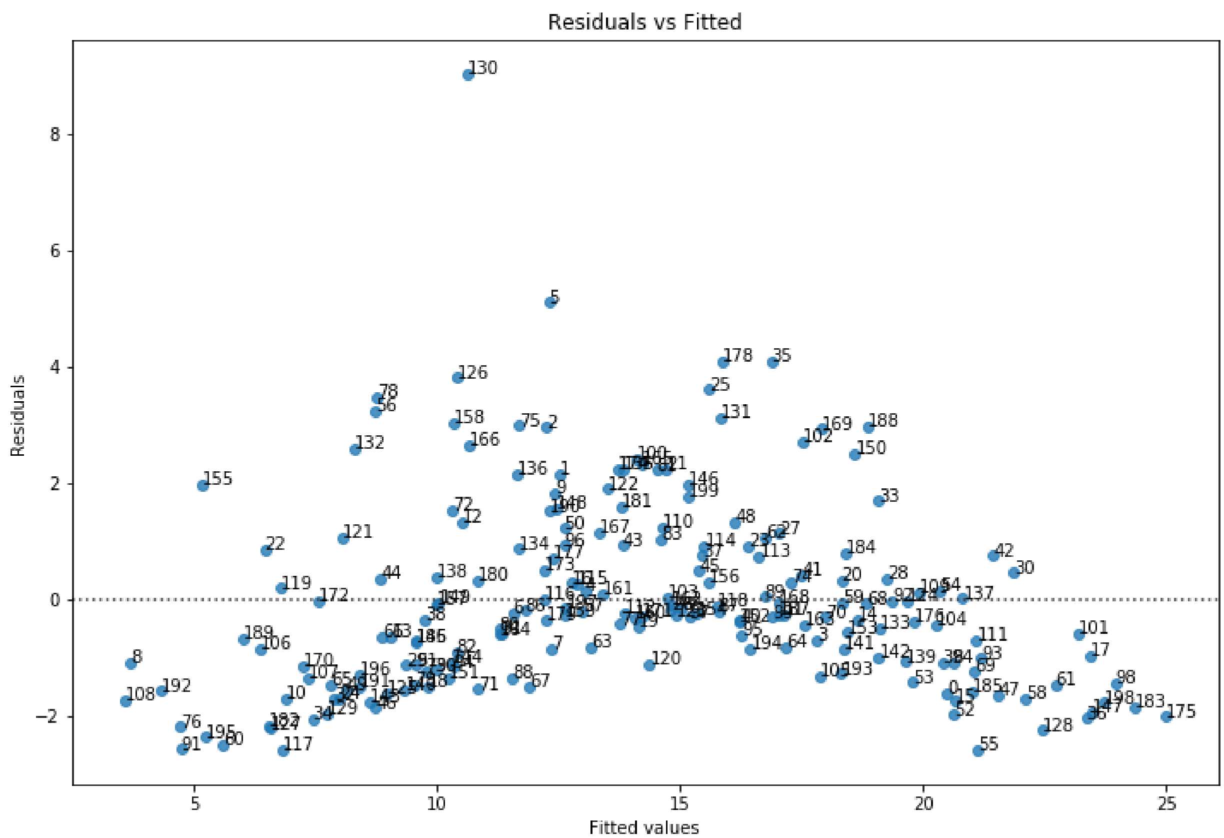
better than before, and no score is larger than 5, but the radio and newspaper still have some collinary like radio and newspaper

1.4 Handling Ourliers (4 pts)

Let's try to identify ourliers from our dataset using residual plot and seaborn package

In [138]:

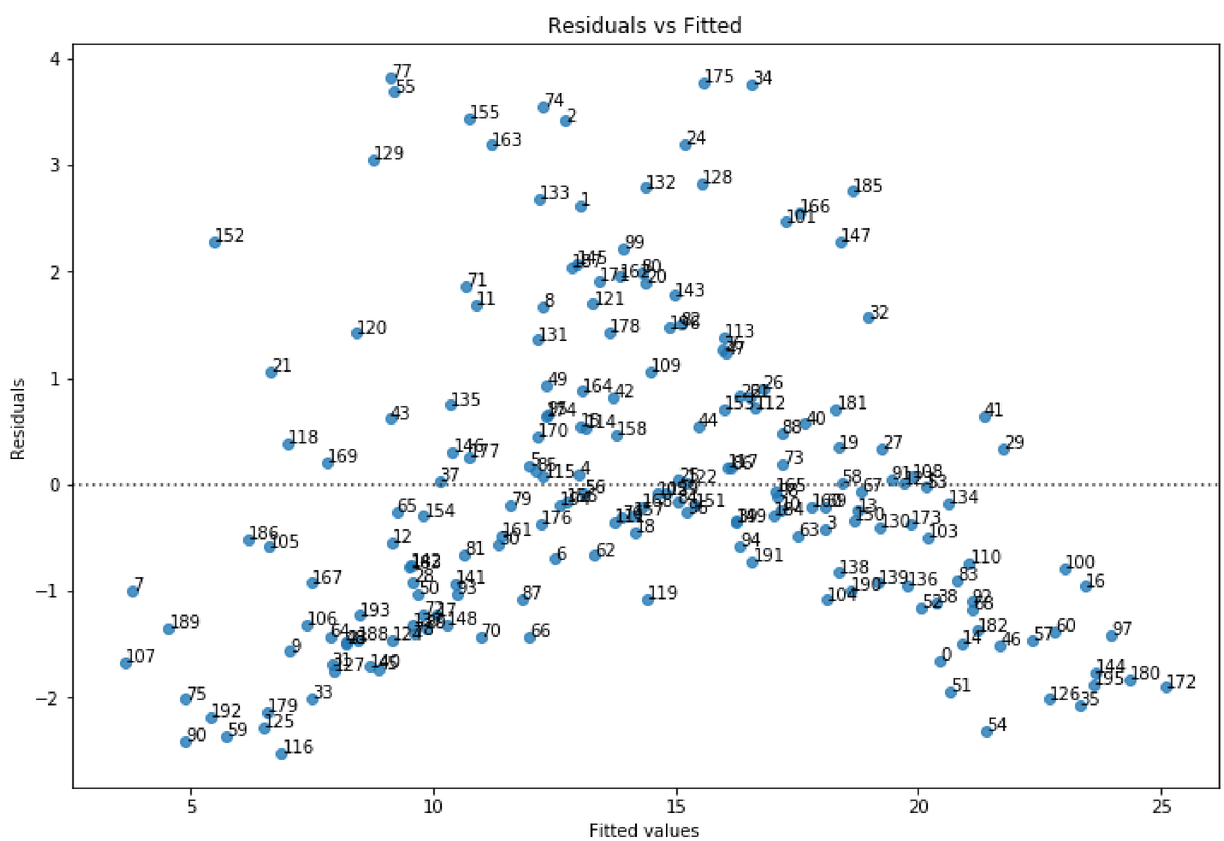
```
import seaborn as sns
import matplotlib.pyplot as plt

## instantiate, fit a linear regression model
model = LinearRegression()
model.fit(X,y)

## compute predicted values and residuals
y_pred = model.predict(X)
y_resid = y_pred-y

plot_lm_1 = plt.figure(1)
plot_lm_1.set_figheight(8)
plot_lm_1.set_figwidth(12)

## use residplot from seaborn to draw the residual plot
plot_lm_1.axes[0]=sns.residplot(y_pred,y_resid)

plot_lm_1.axes[0].set_title('Residuals vs Fitted')
plot_lm_1.axes[0].set_xlabel('Fitted values')
plot_lm_1.axes[0].set_ylabel('Residuals')

# annotations
for i in range(len(y)):
    plot_lm_1.axes[0].annotate(i,
                               xy=(y_pred[i],
                                   y_resid[i]));
```



Question: Find the point with highest residual and remove it from the data. Retrain the model and draw the residual plot again to confirm that the outlier has been removed

In [141]:

```
## remove point with highest residual from our data
mask = ~(y_resid==max(abs(y_resid)))

X = X[mask,:]
y = y[mask]

## train another linear regression model
model = LinearRegression()
model.fit(X,y)
y_pred = model.predict(X)
y_resid = y_pred-y

plot_lm_1 = plt.figure(1)
plot_lm_1.set_figheight(8)
plot_lm_1.set_figwidth(12)

## use residplot from seaborn to draw the residual plot
plot_lm_1.axes[0] = sns.residplot(y_pred,y_resid)

plot_lm_1.axes[0].set_title('Residuals vs Fitted')
plot_lm_1.axes[0].set_xlabel('Fitted values')
plot_lm_1.axes[0].set_ylabel('Residuals')

# annotations
for i in range(len(y)):
    plot_lm_1.axes[0].annotate(i,
                               xy=(y_pred[i],
                                   y_resid[i]));
```



1.5 Linear Regression and GridSearchCV Model Selection in scikit-learn (4 pts)

Let's fit a Linear Regression model with ridge regularization and do model selection to select regularization constant. Fill in the missing code using the hints

In [191]:
```
# follow the usual sklearn pattern: import, instantiate, fit
from sklearn.linear_model import Ridge
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

kcv = KFold(n_splits=5, shuffle=True)
parameters = {'alpha':np.logspace(-3, 3, 7)}
model = Ridge()
score = 0

# Evaluate model by 5 fold cross validation
for train_index, test_index in kcv.split(X, y):
    X_train, X_test = X[train_index],X[test_index]
    y_train, y_test = y[train_index],y[test_index]

#     ## Use GridSearchCV to select regularization constant
    cmodel = GridSearchCV(model,parameters,cv=2,scoring = 'neg_mean_squared_error',
    cmodel.fit(X_train, y_train)
    ## Train Ridge on traning data using the selected regularization constant
    model = Ridge(alpha=cmodel.best_params_['alpha'])
    model.fit(X_train, y_train.ravel())

    score += model.score(X_test, y_test)

print('Model score', score/5)
```

```
Model score 0.9064348030193962
```