In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [2]:
```
df=pd.read_excel('health_data_nulls.xlsx')
df.head()
```

Out[2]:

	Person	Age	Income	Alcohol	Exercise	Smoke	Blood Pressure
0	1	61.0	268300.0	41.0	NaN	3.0	62
1	2	55.0	122200.0	51.0	7.0	56.0	53
2	3	53.0	82100.0	37.0	0.0	55.0	42
3	4	30.0	101400.0	41.0	20.0	61.0	48
4	5	64.0	181100.0	NaN	0.0	70.0	81

In [3]:
```
df=df.dropna()
df.head()
```

Out[3]:

	Person	Age	Income	Alcohol	Exercise	Smoke	Blood Pressure
1	2	55.0	122200.0	51.0	7.0	56.0	53
2	3	53.0	82100.0	37.0	0.0	55.0	42
3	4	30.0	101400.0	41.0	20.0	61.0	48
8	9	59.0	233500.0	25.0	15.0	33.0	66
9	10	44.0	50400.0	64.0	0.0	85.0	54

In [4]:
```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
```

In [5]:
```
y = df['Blood Pressure']
df=df[['Age','Income','Alcohol','Exercise','Smoke']]
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(df, y, test_size=0.2)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

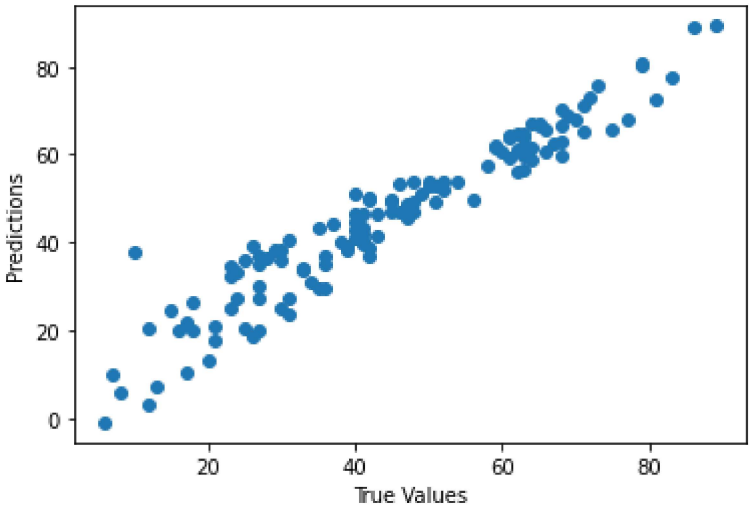
```
(510, 5) (510,)
(128, 5) (128,)
```

In [7]:
```
lm=linear_model.LinearRegression()
model=lm.fit(X_train,y_train)
predictions=lm.predict(X_test)
```

In [8]:
```
predictions[0:5]
```

Out[8]:
```
array([72.635011  , 60.39152806, 33.04744722, -0.98564894, 41.03609251])
```

In [9]:
```
plt.scatter(y_test,predictions)
plt.xlabel("True Values")
plt.ylabel("Predictions")
plt.show()
```



In [10]:
```
print("Score:", model.score(X_test, y_test))
```

```
Score: 0.9137122610008047
```

In [11]:
```
from sklearn.model_selection import KFold
kf = KFold(n_splits=5) #5 splits matches the 80/20 train split
kf.get_n_splits(df) # returns the number of splitting iterations in the cross-validator
print(kf)
KFold(n_splits=5, random_state=None, shuffle=False)
```

```
KFold(n_splits=5, random_state=None, shuffle=False)
```

Out[11]:
```
KFold(n_splits=5, random_state=None, shuffle=False)
```

In [12]:
```
X=df.to_numpy()
Y=y.to_numpy()
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, t_test = Y[train_index], Y[test_index]
```

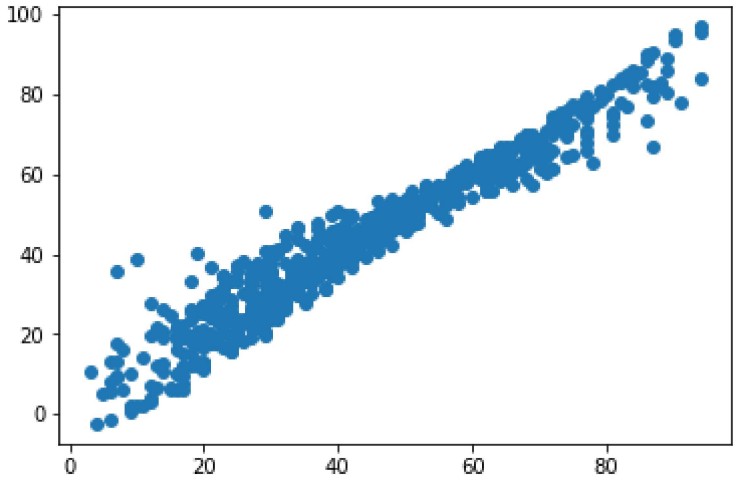
In [13]:
```
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn import metrics
```

In [14]:
```
scores = cross_val_score(model,X, Y, cv=5)
print(scores)
```

```
[0.94136405 0.92254527 0.93177242 0.91735091 0.9503647 ]
```

In [15]:
```
predictions = cross_val_predict(model, X, Y, cv=5)
plt.scatter(Y, predictions)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x1e293ce55b0>
```



In [16]:
```
accuracy = metrics.r2_score(Y,predictions)
accuracy
```

Out[16]:
```
0.933838939927048
```

In [17]:
```
#LOOCV Leave One Out Cross Validation, this is equivalent to split equal to observation
#generally only need to use with small data sets
from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()
loo.get_n_splits(X)


for train_index, test_index in loo.split(X):
    #print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = Y[train_index], Y[test_index]
    #print(X_train, X_test, y_train, y_test)
```

In [18]:
```
#LOOCV is more popular in the classification setting
```

In [20]:
```
mba=pd.ExcelFile('ElecMartSales.xlsx')
df2=mba.parse('Data')
df2.head()
```

Out[20]:

	Date	Day	Time	Region	Card Type	Gender	Buy Category	Items Ordered	Total Cost	High Item	Unnamed: 10	U
0	2016-03-06	Sun	Morning	West	ElecMart	Female	High	4	136.97	79.97	NaN	
1	2016-03-06	Sun	Morning	West	Other	Female	Medium	1	25.55	25.55	NaN	

	Date	Day	Time	Region	Card Type	Gender	Buy Category	Items Ordered	Total Cost	High Item	Unnamed: 10	U
2	2016-03-06	Sun	Afternoon	West	ElecMart	Female	Medium	5	113.95	90.47	NaN	
3	2016-03-06	Sun	Afternoon	NorthEast	Other	Female	Low	1	6.82	6.82	NaN	
4	2016-03-06	Sun	Afternoon	West	ElecMart	Male	Medium	4	147.32	83.21	NaN	

In [21]:
```
two_way=pd.crosstab(df2['Region'], df2['Gender'])
two_way
```

Out[21]:

Gender	Female	Male
Region		
MidWest	43	28
NorthEast	62	53
South	63	30
West	66	55

In [22]:
```
from scipy.stats import chi2_contingency
stat, p, dof, expected = chi2_contingency(two_way)
```

In [23]:
```
stat
```

Out[23]: 5.172525311449733

In [24]:
```
p
```

Out[24]: 0.15959121450450045

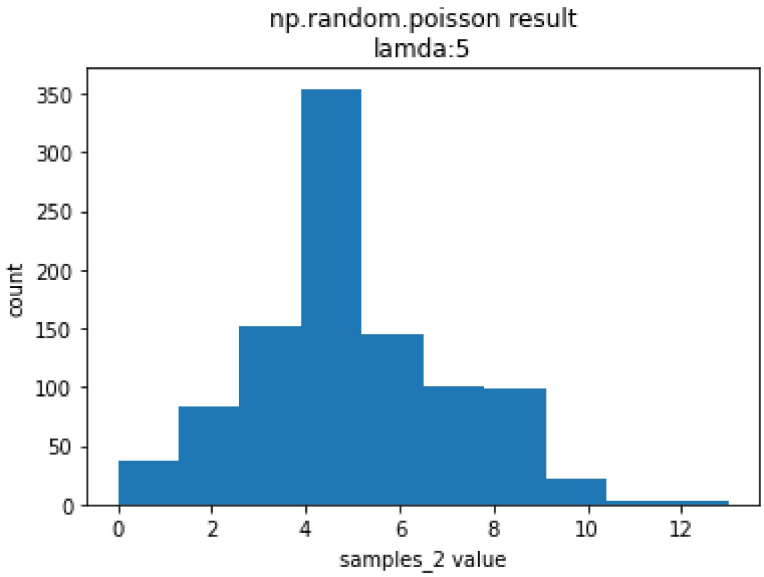
In [25]:
```
from scipy.stats import f_oneway
is_MidWest = df2['Region']=='MidWest'
data1 = df2[is_MidWest]['Total Cost'].to_numpy()
is_NE = df2['Region']=='NorthEast'
data2 = df2[is_NE]['Total Cost'].to_numpy()
is_South = df2['Region']=='South'
data3 = df2[is_South]['Total Cost'].to_numpy()
is_West = df2['Region']=='West'
data4 = df2[is_West]['Total Cost'].to_numpy()
stat, p = f_oneway(data1, data2, data3, data4)
print('stat=%.3f, p=%.3f' % (stat, p))
```

```
stat=1.964, p=0.119
```

In [27]:
```
np.random.seed(0)
lam, size_1, size_2 = 5, 3, 1000
samples_1 = np.random.poisson(lam, size_1)
samples_2 = np.random.poisson(lam, size_2)
answer_1 = abs(np.mean(samples_1) - lam)
answer_2 = abs(np.mean(samples_2) - lam)
print("|Lambda - sample mean| with {} samples is {} and with {} samples is {}. ".format
```

```
|Lambda - sample mean| with 3 samples is 1.666666666666667 and with 1000 samples is 0.05
799999999999983.
```

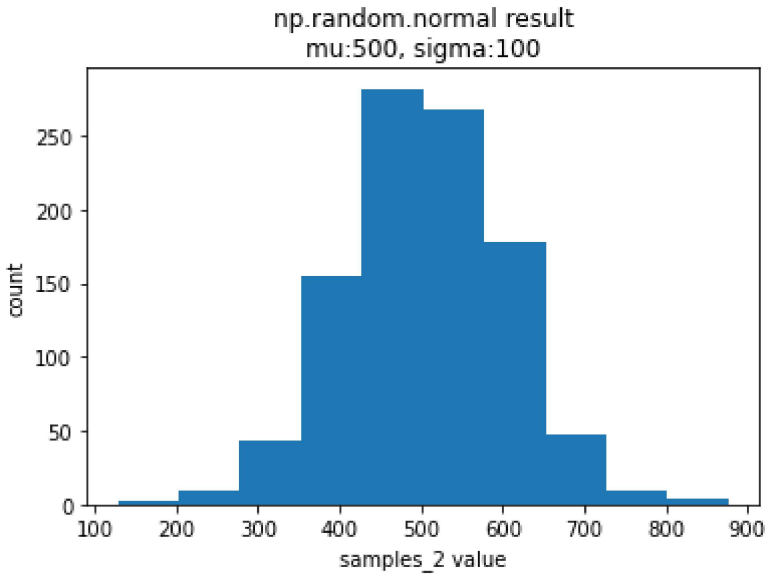
In [28]:
```
plt.hist(samples_2)
plt.xlabel('samples_2 value')
plt.ylabel('count')
plt.title('np.random.poisson result\nlamda:5')
plt.show()
```



In [29]:
```
np.random.seed(99)
mu, sigma, size_1, size_2 = 500, 100, 3, 1000
samples_1 = np.random.normal(mu, sigma, size_1)
samples_2 = np.random.normal(mu, sigma, size_2)
answer_1 = abs(np.mean(samples_1) - mu)
answer_2 = abs(np.mean(samples_2) - mu)
print("|mu - sample mean| with {} samples is {} and with {} samples is {}. ".format(siz
```

```
|mu - sample mean| with 3 samples is 73.2708278661861 and with 1000 samples is 5.6683052
31146178.
```

In [30]:
```
plt.hist(samples_2)
plt.xlabel('samples_2 value')
plt.ylabel('count')
plt.title('np.random.normal result\nmu:500, sigma:100')
plt.show()
```



In [31]:
```
#install wordcloud package
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [33]:
```
df3 = pd.read_csv("winemag-data-130k-v2.csv", index_col=0)
#download from kaggle: https://www.kaggle.com/zynicide/wine-reviews/data
```

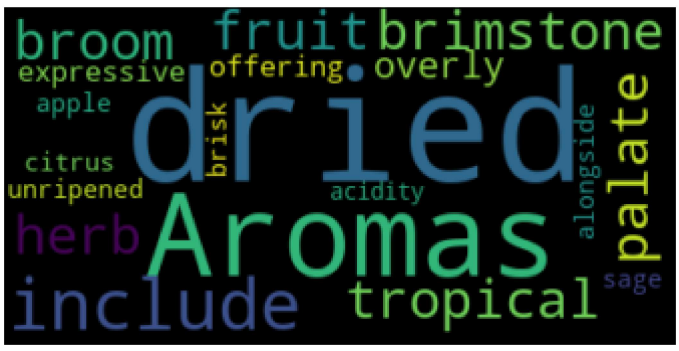
In [34]:
```
text=df3.description[0]
```

In [35]:
```
text
```

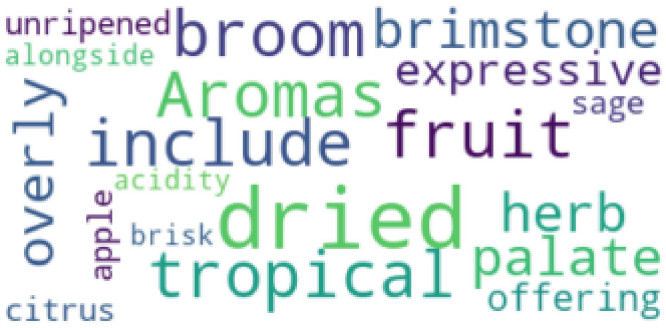
Out[35]: "Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity."

In [36]:
```
wordcloud = WordCloud().generate(text)
```

In [37]:
```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [38]:
```
wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").genera
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



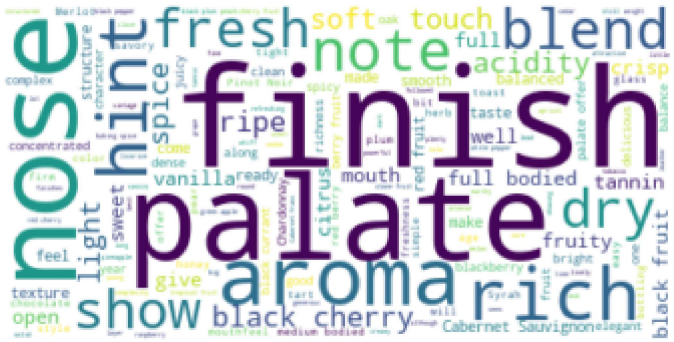
In [39]:
```
text = " ".join(review for review in df3.description)
print ("There are {} words in the combination of all review.".format(len(text)))
```

```
There are 31661073 words in the combination of all review.
```

In [40]:
```
stopwords = set(STOPWORDS)
stopwords.update(["drink", "now", "wine", "flavor", "flavors"])
```

In [41]:
```
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
```

In [42]:
```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



If you want to learn more about making masks for wordclouds (to take on different shapes than the default), check out this site: https://www.datacamp.com/community/tutorials/wordcloud-python