Combined Cycle Power Plant Data Set

The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant. A combined cycle power plant (CCPP) is composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum is colected from and has effect on the Steam Turbine, he other three of the ambient variables effect the GT performance. For comparability with our baseline studies, and to allow 5x2 fold statistical tests be carried out, we provide the data shuffled five times. For each shuffling 2-fold CV is carried out and the resulting 10 measurements are used for statistical testing.

Loading the data

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

In [2]:
```
df = pd.read_excel('Folds5x2_pp.xlsx')
```

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

Out[3]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

Data Analysis

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9568 entries, 0 to 9567
Data columns (total 5 columns):
AT    9568 non-null float64
V     9568 non-null float64
AP    9568 non-null float64
RH    9568 non-null float64
PE    9568 non-null float64
dtypes: float64(5)
memory usage: 373.8 KB
```

In [9]:
```
df.isnull().any()
```

Out[9]:
```
AT    False
V     False
AP    False
RH    False
PE    False
dtype: bool
```

or

In [11]:
```
df.isnull().any().any()
```

Out[11]:
```
False
```

or

In [13]:
```
df.isnull().T.any().T.any()
```

Out[13]:
```
False
```

In [14]:
```
df.describe()
```

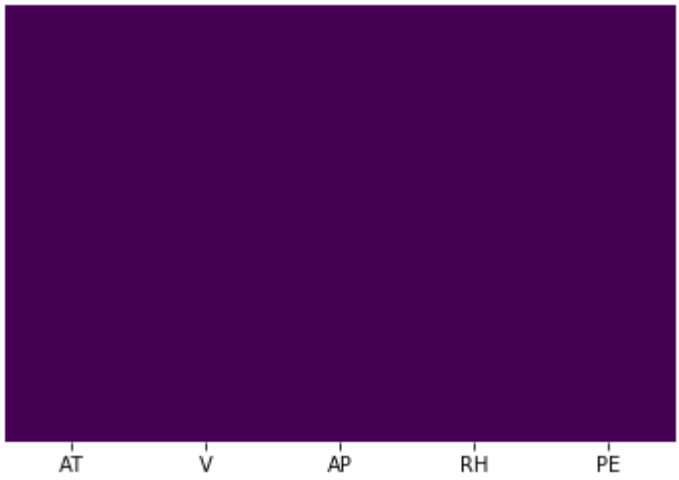
Out[14]:

	AT	V	AP	RH	PE
count	9568.000000	9568.000000	9568.000000	9568.000000	9568.000000
mean	19.651231	54.305804	1013.259078	73.308978	454.365009
std	7.452473	12.707893	5.938784	14.600269	17.066995
min	1.810000	25.360000	992.890000	25.560000	420.260000
25%	13.510000	41.740000	1009.100000	63.327500	439.750000
50%	20.345000	52.080000	1012.940000	74.975000	451.550000
75%	25.720000	66.540000	1017.260000	84.830000	468.430000
max	37.110000	81.560000	1033.300000	100.160000	495.760000

Checking for null in Dataset

In [28]:
```
sns.heatmap(df.isnull(), cbar=False, cmap='viridis', yticklabels=False)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x11211438>



Getting the correlation Metrics

In [29]:
```
df.corr()
```

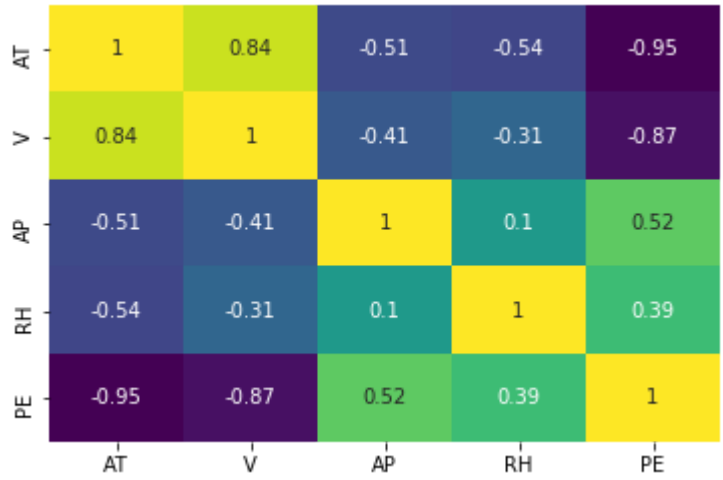
Out[29]:

	AT	V	AP	RH	PE
AT	1.000000	0.844107	-0.507549	-0.542535	-0.948128
V	0.844107	1.000000	-0.413502	-0.312187	-0.869780
AP	-0.507549	-0.413502	1.000000	0.099574	0.518429
RH	-0.542535	-0.312187	0.099574	1.000000	0.389794
PE	-0.948128	-0.869780	0.518429	0.389794	1.000000

Lets get it in a visual form, that will be more helpful to observer

In [30]:
```
sns.heatmap(df.corr(), annot=True, cmap='viridis', cbar=False)
```

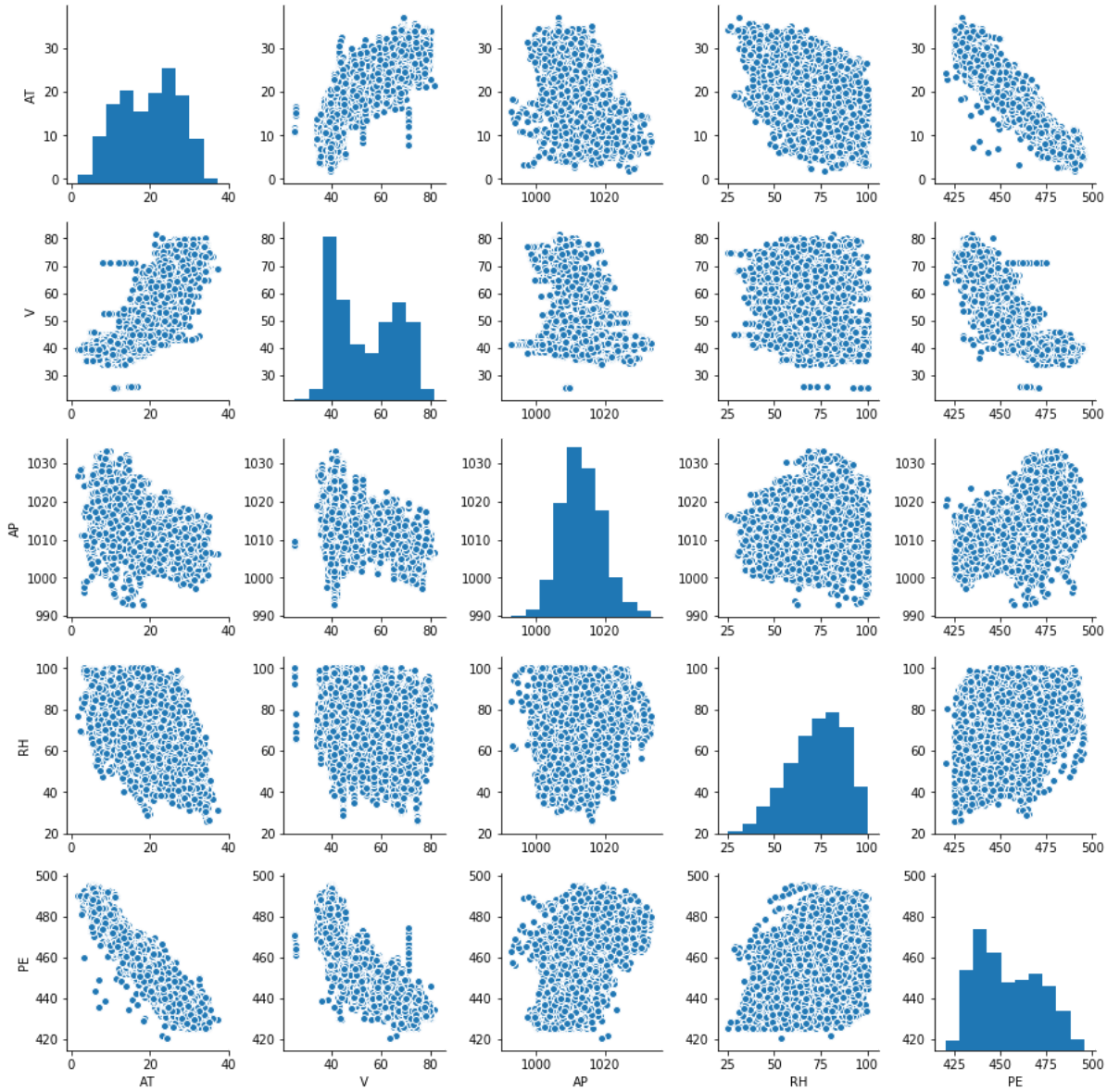
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x110912b0>



Exploratory Data Analysis

In [15]:
```
sns.pairplot(df)
```

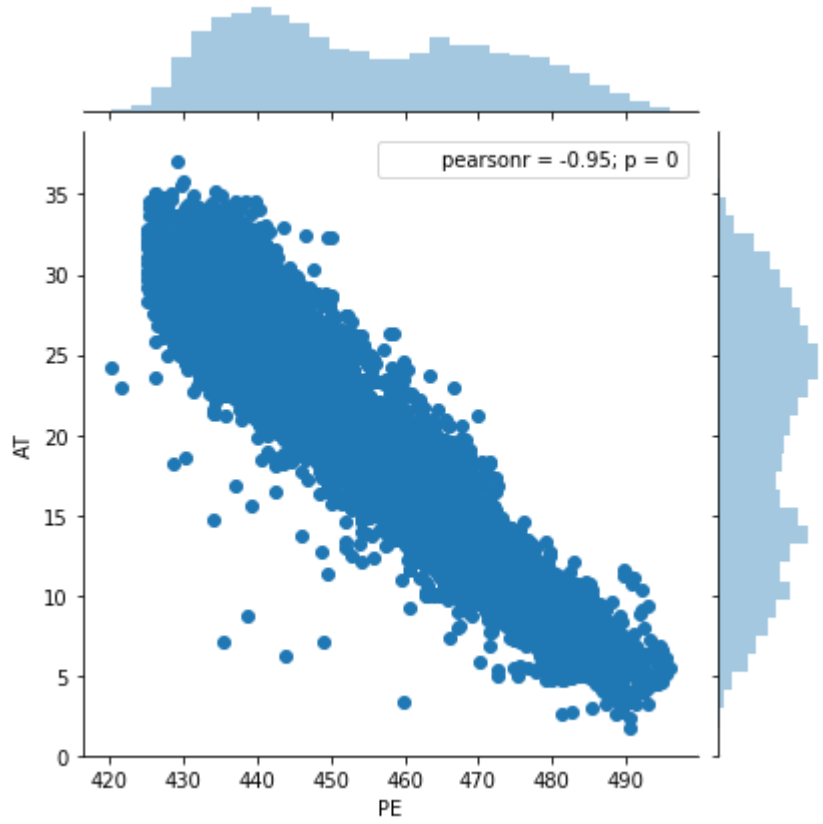
Out[15]: <seaborn.axisgrid.PairGrid at 0xcb23198>



Looks like the Temperature(AT) has some kind of linear relation ship with Power Output (PE)

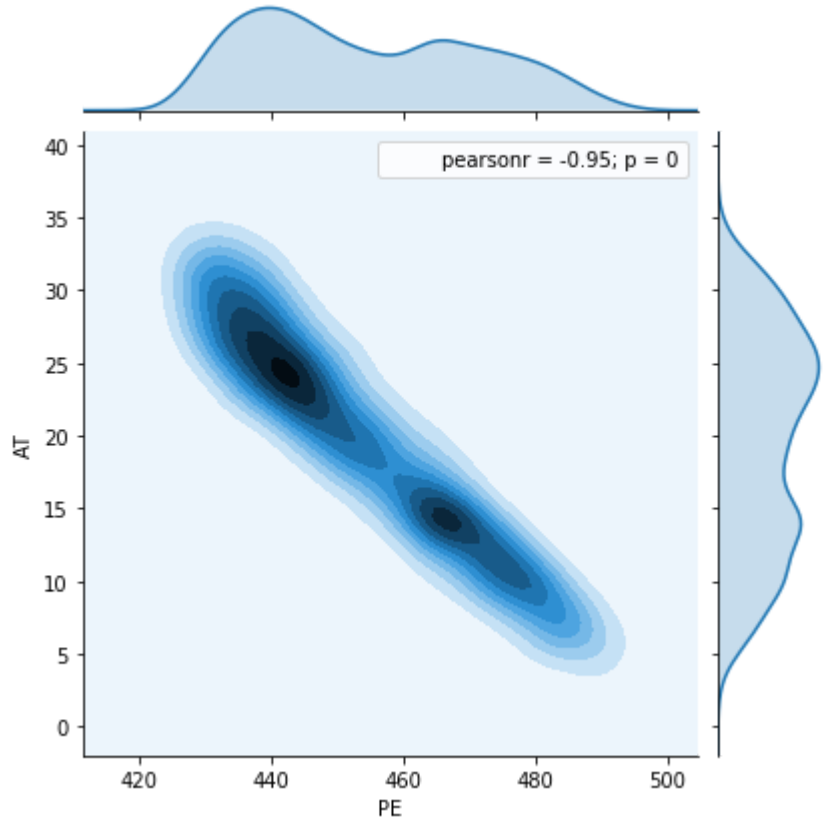
In [20]:
```
sns.jointplot(df.PE, df.AT, data=df )
```

Out[20]: <seaborn.axisgrid.JointGrid at 0xf064ac8>



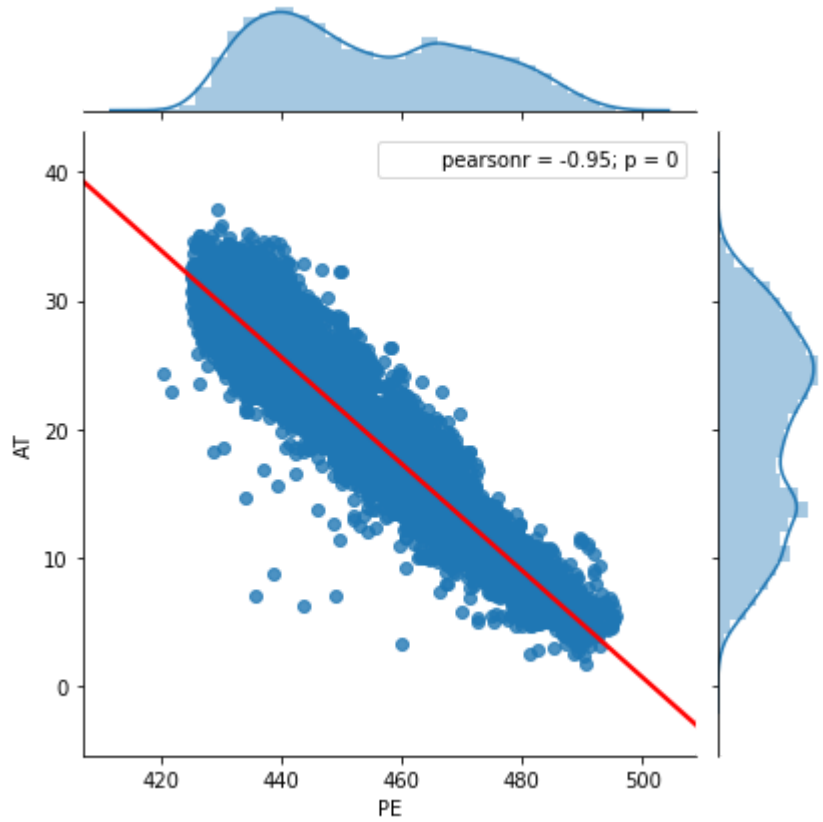
In [43]:
```
sns.jointplot(df.PE, df.AT, df, kind='kde')
```

Out[43]: <seaborn.axisgrid.JointGrid at 0x105be898>



In [47]:
```
sns.jointplot(df.PE, df.AT, df, kind='regression', joint_kws={'line_kws':{'color':'red'}})
```

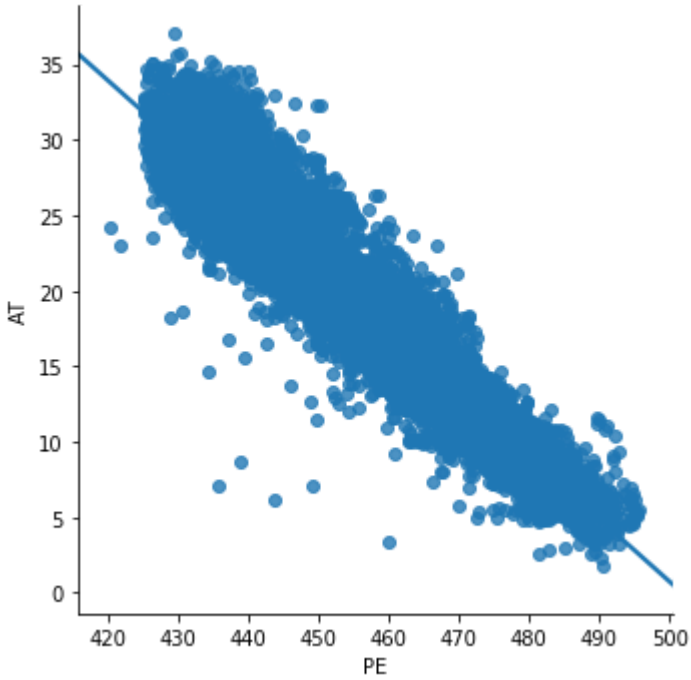
Out[47]: <seaborn.axisgrid.JointGrid at 0x131db160>



Or the above Regression line can also be drawn using seaborn lmplot

In [23]:
```
sns.lmplot(x='PE', y='AT', data=df)
```

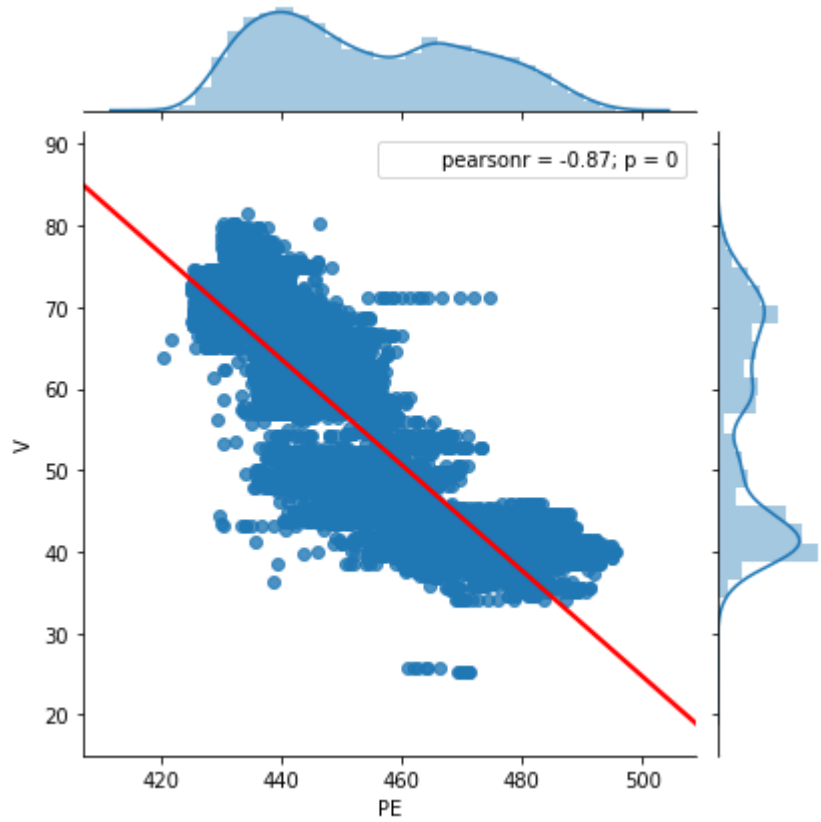
Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0xf4dd588>
```



Also Exhaust Vaccume (V) shows some kind of lineat relationship with Power Output (PE)

In [50]:
```
sns.jointplot(df.PE, df.V, df, kind='regression',joint_kws={'line_kws':{'colo
r':'red'}})
```

Out[50]: <seaborn.axisgrid.JointGrid at 0x146bf518>



Lets see what regression line we get for the above graph

Training & Test Data sets

Using sklearn model selection

In [51]:
```
from sklearn.model_selection import train_test_split
```

In [52]:
```
X_train, X_test, y_train, y_test = train_test_split(df.drop('PE', axis=1), df[
'PE'], test_size=0.3, random_state=101)
```

Training & Testing the model

In [53]:
```
from sklearn.linear_model import LinearRegression
```

In [54]:
```
lm = LinearRegression()
```

In [55]:
```
lm.fit(X_train, y_train)
```

Out[55]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Predicting Test Data

Now that we have fit our model, let's evaluate its performance by predicting off the test values!

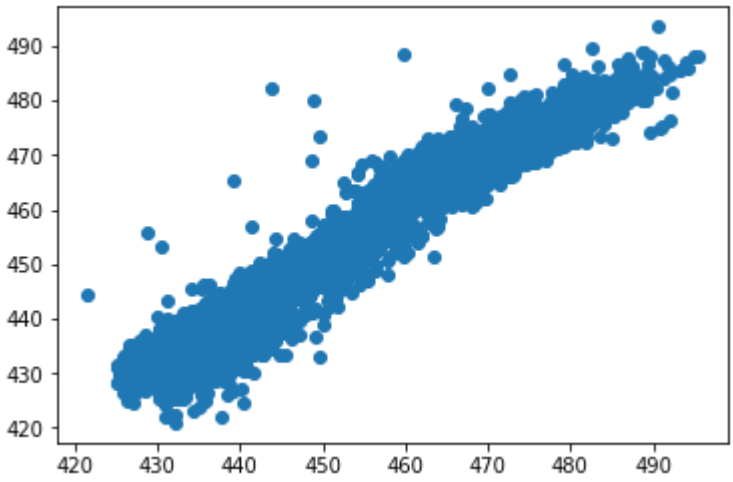
Use lm.predict() to predict off the X_test set of the data.

In [56]:
```
pred = lm.predict(X_test)
```

Lets see how well our model does on predictions visually

In [57]:
```
plt.scatter(y_test, pred)
```

Out[57]:
```
<matplotlib.collections.PathCollection at 0x14bb5f60>
```



Looks like the model does a pretty good job !!!

Evaluating the model

Now lets evaluate the model using metrics and R2 Score

In [58]:
```
from sklearn import metrics
```

R2 Score for the model

In [59]:
```
print(metrics.r2_score(y_test, pred))
```

```
0.925759853813
```

That's quite a good score!

Errors for the model

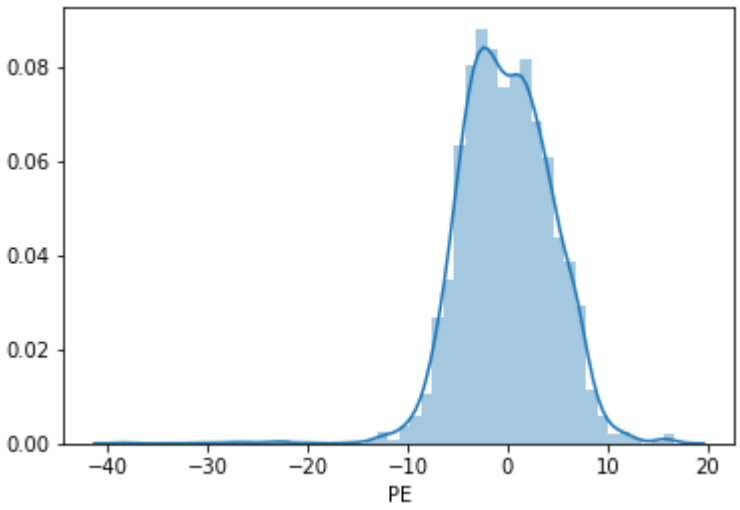
In [60]:
```
print('MAE: ', metrics.mean_absolute_error(y_test, pred))
print('MSE: ', metrics.mean_squared_error(y_test, pred))
print('RMSE: ', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE:  3.60311980328
MSE:  21.023342493
RMSE:  4.5851218624
```

Let's check out the distribution of the residuals

In [61]:
```
sns.distplot((y_test-pred))
```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x150a5630>



Coefficients of PE

In [63]:
```
coefficients = pd.DataFrame(lm.coef_, index=df.columns[:-1], columns=['Coeffic
ients'])
```

In [64]:
```
coefficients
```

Out[64]:

	Coefficients
AT	-1.981308
V	-0.233927
AP	0.061984
RH	-0.158979

Conclusions

- Holding all other features constant, a 1 unit increase in AT is associated with a decrease of 1.981 unit in PE
- Holding all other features constant, a 1 unit increase in V is associated with a decrease of 0.2339 unit in PE
- Holding all other features constant, a 1 unit increase in AP is associated with a increase of 0.06198 unit in PE
- Holding all other features constant, a 1 unit increase in RH is associated with a decrease of 0.1589 unit in PE