In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
import re
import lmdiag
```

In [2]:
```
data = pd.read_csv('cars.csv')
data.head(3)
```

Out[2]:

	speed	dist
0	4	2
1	4	10
2	7	4

In [3]:
```
data.info()
```

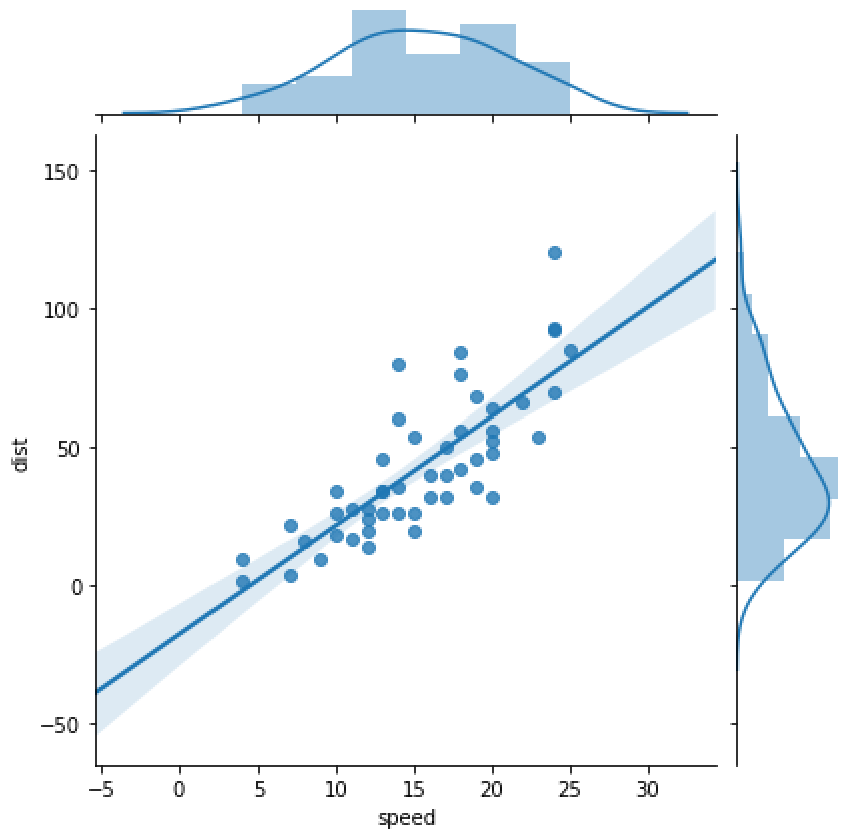
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 2 columns):
speed    50 non-null int64
dist     50 non-null int64
dtypes: int64(2)
memory usage: 928.0 bytes
```

In [4]:
```
data.corr()
```

Out[4]:

	speed	dist
speed	1.000000	0.806895
dist	0.806895	1.000000

In [5]:
```
sns.jointplot(data['speed'],data['dist'],kind='reg')
plt.show()
```



$$slope = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x}^2)}$$

$$y - intercept = \bar{y} - m\bar{x}$$

In [6]:
```
Xbar = data['speed'].mean()
Ybar = data['dist'].mean()
```

In [7]:
```
data['X-Xbar'] = data['speed'] - Xbar
data['Y-Ybar'] = data['dist'] - Ybar
data['(X-Xbar)(Y-Ybar)'] = data['X-Xbar']*data['Y-Ybar']
data['(X-Xbar)sqr'] = data['X-Xbar']**2
```

In [8]:
```
data.head()
```

Out[8]:

	speed	dist	X-Xbar	Y-Ybar	(X-Xbar)(Y-Ybar)	(X-Xbar)sqr
0	4	2	-11.4	-40.98	467.172	129.96
1	4	10	-11.4	-32.98	375.972	129.96
2	7	4	-8.4	-38.98	327.432	70.56
3	7	22	-8.4	-20.98	176.232	70.56
4	8	16	-7.4	-26.98	199.652	54.76

In [9]:
```
slope_nume = data['(X-Xbar)(Y-Ybar)'].sum()
slope_denom = data['(X-Xbar)sqr'].sum()
slope = slope_nume/slope_denom
slope
```

Out[9]:
```
3.932408759124088
```

In [10]:
```
y_intercept = Ybar - (slope*Xbar)
y_intercept
```

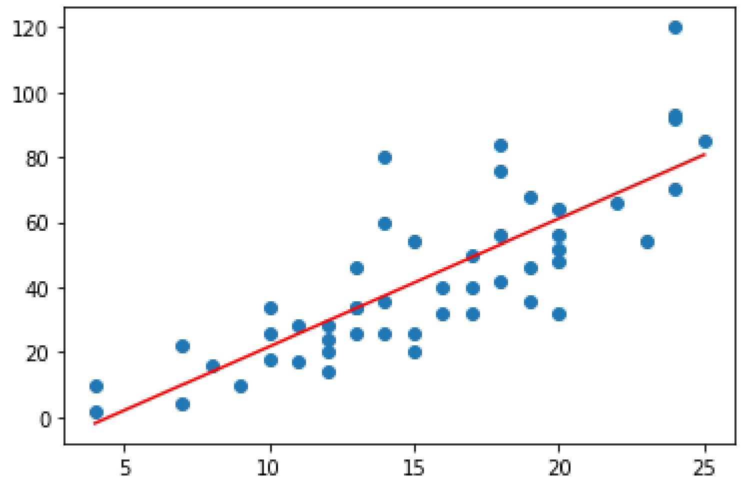
Out[10]:
```
-17.57909489051096
```

In [11]:
```
data['Y-pred'] = (slope * data['speed']) + y_intercept
data.head()
```

Out[11]:

	speed	dist	X-Xbar	Y-Ybar	(X-Xbar)(Y-Ybar)	(X-Xbar)sqr	Y-pred
0	4	2	-11.4	-40.98	467.172	129.96	-1.849460
1	4	10	-11.4	-32.98	375.972	129.96	-1.849460
2	7	4	-8.4	-38.98	327.432	70.56	9.947766
3	7	22	-8.4	-20.98	176.232	70.56	9.947766
4	8	16	-7.4	-26.98	199.652	54.76	13.880175

In [12]:
```
plt.scatter(data['speed'],data['dist'])
plt.plot(data['speed'],data['Y-pred'],'r')
plt.show()
```



In [13]:
```
data['Y- Ypredsqr'] = (data['dist'] - data['Y-pred'])**2
```

In [14]:
```
mse = data['Y- Ypredsqr'].mean()
print(mse)
rmse = math.sqrt(mse)
print(rmse)
```

```
227.07042102189777
15.068855995791377
```

Model Building

In [15]:
```
import statsmodels.formula.api as smf
model = smf.ols(formula='dist~speed',data = data).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   dist   R-squared:                       0.651
Model:                            OLS   Adj. R-squared:                  0.644
Method:                 Least Squares   F-statistic:                     89.57
Date:                Sun, 09 Feb 2020   Prob (F-statistic):           1.49e-12
Time:                        18:26:34   Log-Likelihood:                -206.58
No. Observations:                  50   AIC:                             417.2
Df Residuals:                      48   BIC:                             421.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -17.5791      6.758     -2.601      0.012     -31.168      -3.990
speed          3.9324      0.416      9.464      0.000       3.097       4.768
==============================================================================
Omnibus:                        8.975   Durbin-Watson:                   1.676
Prob(Omnibus):                  0.011   Jarque-Bera (JB):                8.189
Skew:                           0.885   Prob(JB):                       0.0167
Kurtosis:                       3.893   Cond. No.                         50.7
==============================================================================

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

In [16]:
```
data['model_pred'] = model.predict(data['speed'])
data.head()
```

Out[16]:

	speed	dist	X-Xbar	Y-Ybar	(X-Xbar)(Y-Ybar)	(X-Xbar)sqr	Y-pred	Y- Ypredsqr	model_pred
0	4	2	-11.4	-40.98	467.172	129.96	-1.849460	14.818341	-1.849460
1	4	10	-11.4	-32.98	375.972	129.96	-1.849460	140.409699	-1.849460
2	7	4	-8.4	-38.98	327.432	70.56	9.947766	35.375925	9.947766
3	7	22	-8.4	-20.98	176.232	70.56	9.947766	145.256334	9.947766
4	8	16	-7.4	-26.98	199.652	54.76	13.880175	4.493657	13.880175

In [17]:
```
math.sqrt(sum((model.resid)**2)/len(data))
```

Out[17]: 15.068855995791377

In [18]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [19]:
```
mean_squared_error(data['dist'],data['model_pred'])
```

Out[19]: 227.07042102189777

In [20]:
```
mean_absolute_error(data['dist'],data['model_pred'])
```

Out[20]: 11.580119124087592

In [21]:
```
# Method 2
import statsmodels.api as sm
```

In [22]:
```
X = data['speed']
Y = data['dist']
```

In [23]:
```
X = sm.add_constant(X)
model1 = sm.OLS(Y,X).fit()
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   dist   R-squared:                       0.651
Model:                            OLS   Adj. R-squared:                  0.644
Method:                 Least Squares   F-statistic:                     89.57
Date:                Sun, 09 Feb 2020   Prob (F-statistic):           1.49e-12
Time:                        18:26:34   Log-Likelihood:                -206.58
No. Observations:                  50   AIC:                             417.2
Df Residuals:                      48   BIC:                             421.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -17.5791      6.758     -2.601      0.012     -31.168      -3.990
speed          3.9324      0.416      9.464      0.000       3.097       4.768
==============================================================================
Omnibus:                        8.975   Durbin-Watson:                   1.676
Prob(Omnibus):                  0.011   Jarque-Bera (JB):                8.189
Skew:                           0.885   Prob(JB):                       0.0167
Kurtosis:                       3.893   Cond. No.                         50.7
==============================================================================

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

C:\Users\admin\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning: Method .ptp is deprecated and
will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

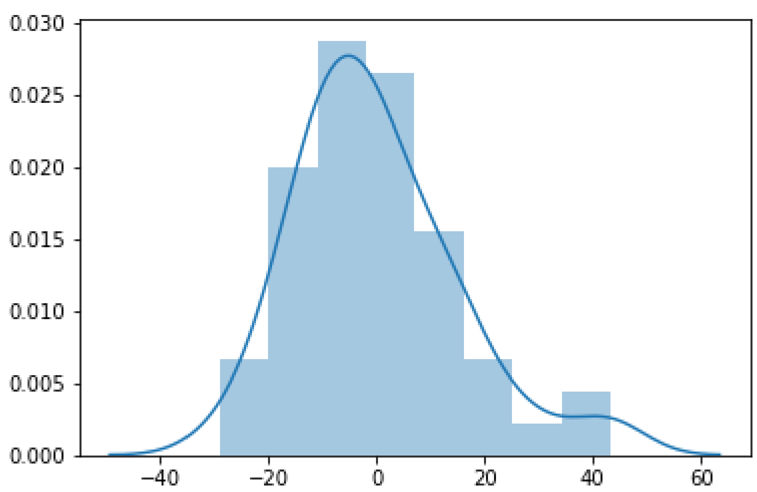
In [ ]:

In [24]:
```
# Method 3
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
model2 = LR.fit(X,Y)
model2.predict(X)
```

Out[24]:
```
array([-1.84945985, -1.84945985,  9.94776642,  9.94776642, 13.88017518,
       17.81258394, 21.7449927 , 21.7449927 , 21.7449927 , 25.67740146,
       25.67740146, 29.60981022, 29.60981022, 29.60981022, 29.60981022,
       33.54221898, 33.54221898, 33.54221898, 33.54221898, 37.47462774,
       37.47462774, 37.47462774, 37.47462774, 41.4070365 , 41.4070365 ,
       41.4070365 , 45.33944526, 45.33944526, 49.27185401, 49.27185401,
       49.27185401, 53.20426277, 53.20426277, 53.20426277, 53.20426277,
       57.13667153, 57.13667153, 57.13667153, 61.06908029, 61.06908029,
       61.06908029, 61.06908029, 61.06908029, 68.93389781, 72.86630657,
       76.79871533, 76.79871533, 76.79871533, 76.79871533, 80.73112409])
```

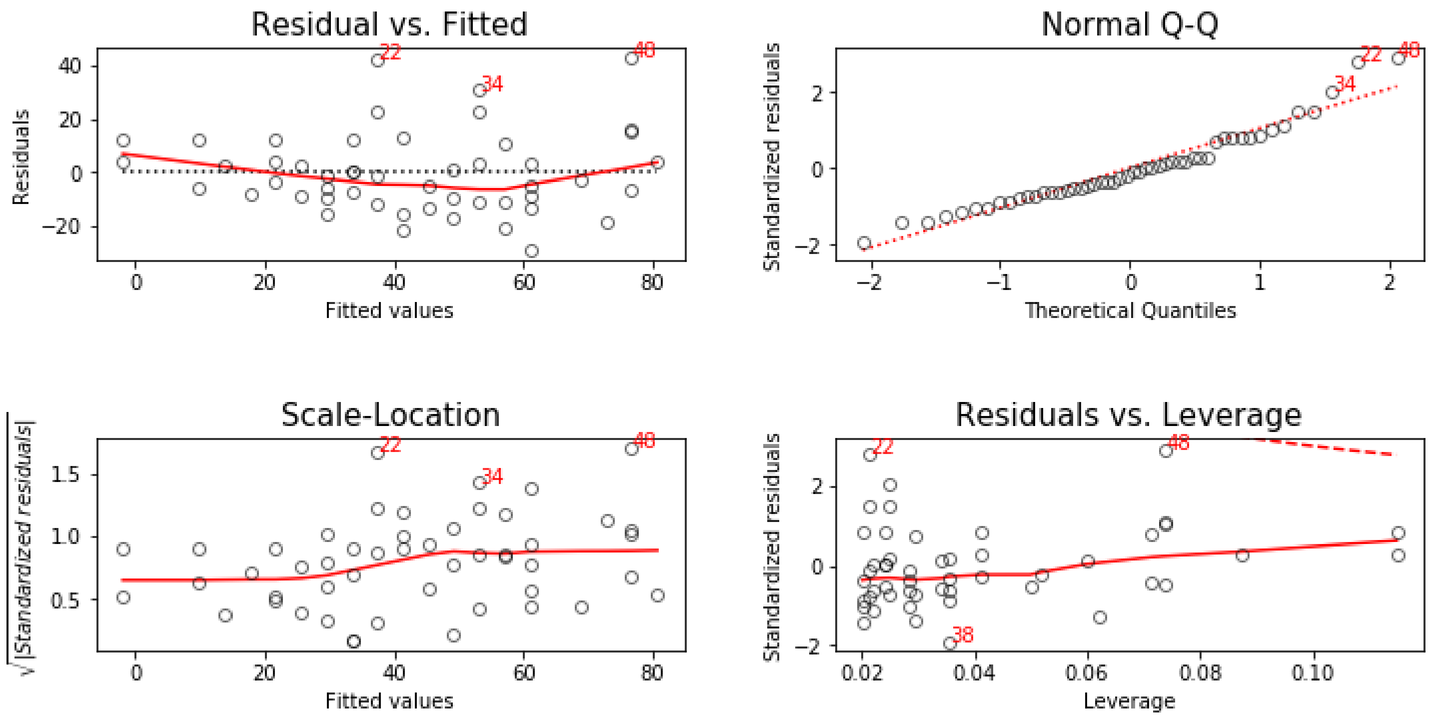
Residual Analysis

In [25]:
```
sns.distplot(model.resid)
plt.savefig('dist.png')
plt.show()
```

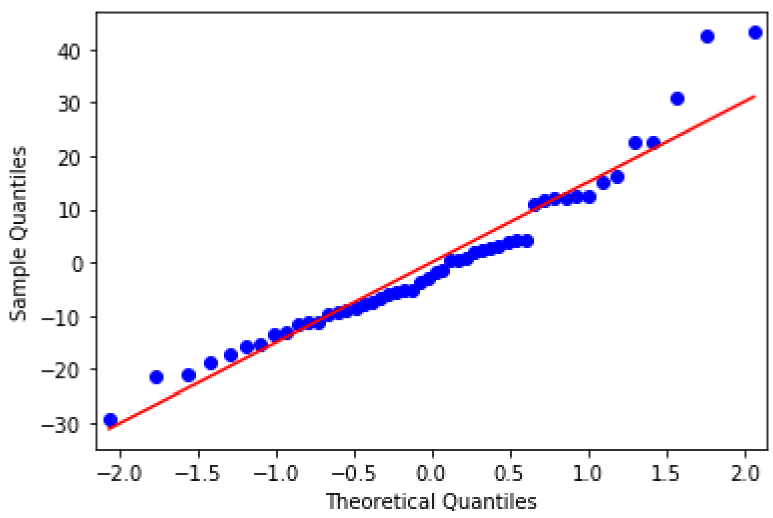


```
In [26]: plt.figure(figsize=(10,5))
         lmdiag.plot(model)
```

```
Out[26]: <module 'matplotlib.pyplot' from 'C:\\Users\\admin\\Anaconda3\\lib\\site-packages\\matplotlib\\pyplot.py'>
```




```
In [27]: sm.graphics.qqplot(model.resid,line='s')
         plt.savefig('qqplot.png')
         plt.show()
```



Hypothesis Testing with Shapiro Wilk Test

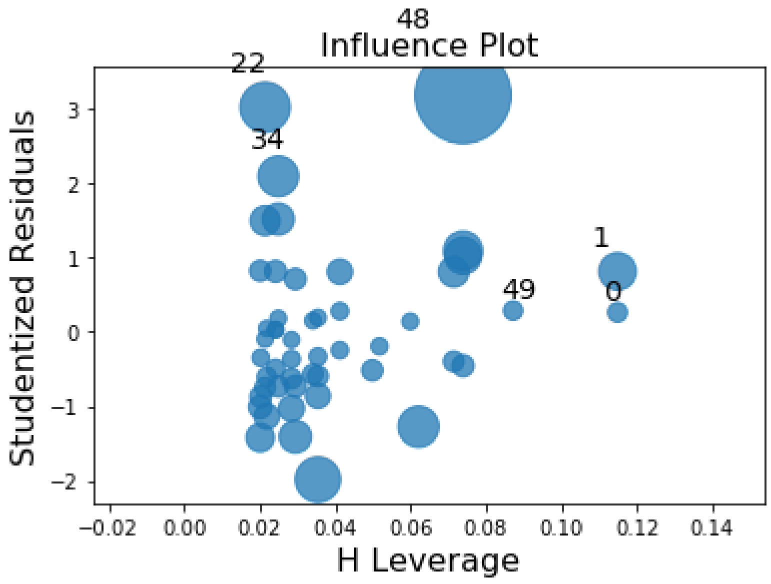
- Null Hypothesis - If pvalue > 0.05 then accept the model
- Alternate Hypothesis - If pvalue < 0.05 then reject the model

```
In [28]: from scipy.stats import shapiro
         shapiro(model.resid)
```

```
Out[28]: (0.9450905919075012, 0.021524606272578 24)
```

From above Pvalue we are rejecting the Null Hypothesis

In [29]:
```
sm.graphics.influence_plot(model)
plt.show()
```



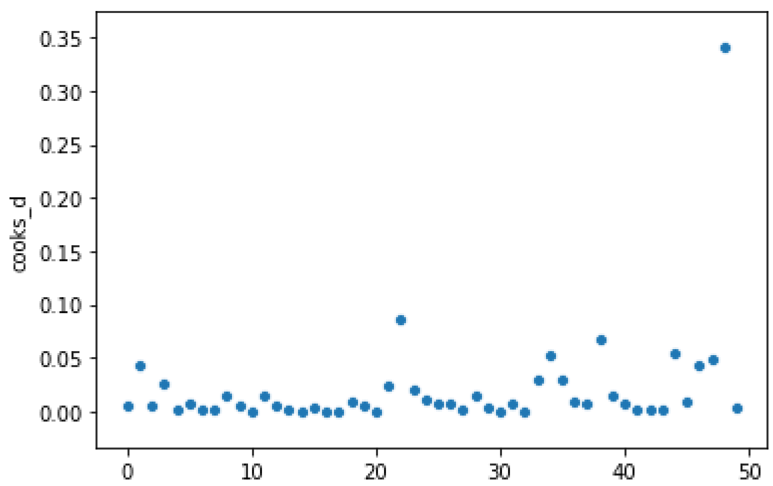
In [30]:
```
influ = model.get_influence()
cooks = influ.summary_frame()
cooks.head()
```

Out[30]:

	dfb_Intercept	dfb_speed	cooks_d	standard_resid	hat_diag	dffits_internal	student_resid	dffits
0	0.094402	-0.086246	0.004592	0.266042	0.114861	0.095836	0.263450	0.094903
1	0.292425	-0.267160	0.043514	0.818933	0.114861	0.295005	0.816078	0.293977
2	-0.107498	0.093693	0.006202	-0.401346	0.071504	-0.111376	-0.397812	-0.110396
3	0.218976	-0.190855	0.025467	0.813266	0.071504	0.225687	0.810353	0.224879
4	0.034075	-0.029014	0.000645	0.142162	0.059971	0.035907	0.140703	0.035539

In [31]:
```
sns.scatterplot(cooks.index,cooks.cooks_d,data=cooks)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x21b05be9688>



In [32]:
```
cooks[cooks['cooks_d']>0.08]
```

Out[32]:

	dfb_Intercept	dfb_speed	cooks_d	standard_resid	hat_diag	dffits_internal	student_resid	dffits
22	0.248506	-0.115581	0.085552	2.795166	0.021431	0.413647	3.022829	0.447338
48	-0.577473	0.769020	0.340396	2.919060	0.073985	0.825101	3.184993	0.900270

From above we can say that row no.s 22 and 48 are considered outliers. So remove those from data

In [33]:
```
outliers = cooks[cooks['cooks_d']>0.08].index
new_data = data.drop(labels=outliers)
data.shape,new_data.shape
new_data.head()
```

Out[33]:

	speed	dist	X-Xbar	Y-Ybar	(X-Xbar)(Y-Ybar)	(X-Xbar)sqr	Y-pred	Y- Ypredsqr	model_pred
0	4	2	-11.4	-40.98	467.172	129.96	-1.849460	14.818341	-1.849460
1	4	10	-11.4	-32.98	375.972	129.96	-1.849460	140.409699	-1.849460
2	7	4	-8.4	-38.98	327.432	70.56	9.947766	35.375925	9.947766
3	7	22	-8.4	-20.98	176.232	70.56	9.947766	145.256334	9.947766
4	8	16	-7.4	-26.98	199.652	54.76	13.880175	4.493657	13.880175

smf model

In [34]:
```
new_model = smf.ols(formula='dist~speed',data=new_data).fit()
print(new_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   dist   R-squared:                       0.702
Model:                            OLS   Adj. R-squared:                  0.695
Method:                 Least Squares   F-statistic:                     108.3
Date:                Sun, 09 Feb 2020   Prob (F-statistic):           1.13e-13
Time:                        18:26:37   Log-Likelihood:                -189.16
No. Observations:                  48   AIC:                             382.3
Df Residuals:                      46   BIC:                             386.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -15.5336      5.699     -2.726      0.009     -27.005      -4.062
speed          3.6812      0.354     10.405      0.000       2.969       4.393
==============================================================================
Omnibus:                        2.282   Durbin-Watson:                   1.907
Prob(Omnibus):                  0.320   Jarque-Bera (JB):                1.837
Skew:                           0.479   Prob(JB):                        0.399
Kurtosis:                       2.968   Cond. No.                         50.2
==============================================================================

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

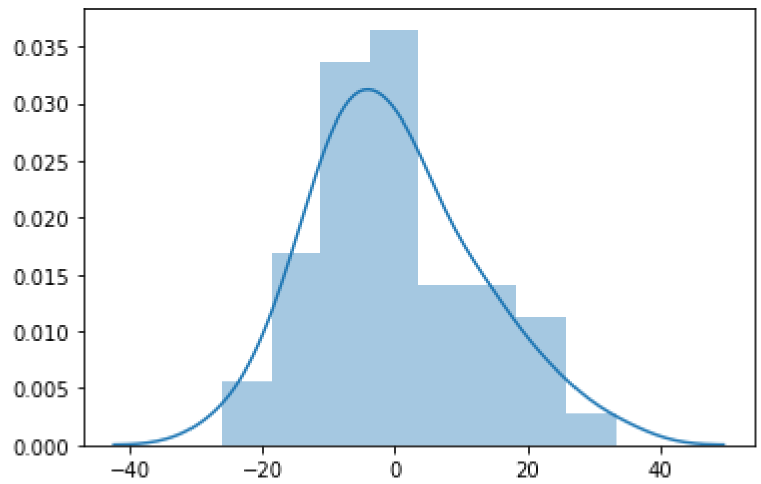
In [35]:
```
new_y_pred = new_model.predict(new_data['speed'])
```

In [36]:
```
new_mse = mean_squared_error(new_data['dist'],new_y_pred)
```

In [37]:
```
math.sqrt(new_mse)
```

Out[37]: 12.453279858508711

In [38]:
```
sns.distplot(new_model.resid)
plt.savefig('finaldist.png')
plt.show()
```



In [39]:
```
shapiro(new_model.resid)
```

Out[39]: (0.9790715575218201, 0.5406291484832764)