Lab 6&7 ¶

In [ ]:

```
#n01496099
#Vishal Kumar
```

Please submit as a pdf.

Submission deadline is mentioend in the blackbloard submission system.

Imports

** Import pandas, numpy, matplotlib,and seaborn. Then set %matplotlib inline (You'll import sklearn as you need it.)**

In [62]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
import math

import warnings
warnings.filterwarnings("ignore")
```

Import Dataset

Datafile Name: Enrollment Forecast

Number of cases: 29 Variable Names:

- 1.YEAR: 1961 = 1, 1989 = 29
- 2.ROLL: Fall undergraduate enrollment
- 3.UNEM: January unemployment rate (%) for New Mexico
- 4.HGRAD: Spring high schoolgraduates in New Mexico
- 5.INC: Per capita income in Albuquerque (1961 dollars)

In [63]:

```
df_data=pd.read_csv('C:/Users/Vishal Kumar/Desktop/ITS/2nd Semester/Introduction to Data An
# reading the dataset
```

In [64]:

```
#df_data=pd.read_csv('C:/Users/Vishal Kumar/Desktop/ITS/2nd Semester/Introduction to Data A
#df_data.head()
```

In [65]:

```
# Write your code here to generate the following output done
df_data.head()
```

Out[65]:

	year	roll	unem	hgrad	inc
0	1	5501	8.1	9552	1923
1	2	5945	7.0	9680	1961
2	3	6629	7.3	9731	1979
3	4	7556	7.5	11666	2030
4	5	8716	7.0	14675	2112

In [ ]:

```

```

Check the head of customers, and check out its info() and describe() methods.

In [66]:

```
# Write your code here to generate the following output
df_data.head()
```

Out[66]:

	year	roll	unem	hgrad	inc
0	1	5501	8.1	9552	1923
1	2	5945	7.0	9680	1961
2	3	6629	7.3	9731	1979
3	4	7556	7.5	11666	2030
4	5	8716	7.0	14675	2112

In [67]:

```
find=df_data[['year','roll','unem','hgrad','inc']]
```

In [68]:

```
# Write your code here to generate the following output
find.describe()
```

Out[68]:

	year	roll	unem	hgrad	inc
count	29.000000	29.000000	29.000000	29.000000	29.000000
mean	15.000000	12707.034483	7.717241	16528.137931	2729.482759
std	8.514693	3254.076987	1.123155	2926.926676	461.429194
min	1.000000	5501.000000	5.700000	9552.000000	1923.000000
25%	8.000000	10167.000000	7.000000	15723.000000	2351.000000
50%	15.000000	14395.000000	7.500000	17203.000000	2863.000000
75%	22.000000	14969.000000	8.200000	18266.000000	3127.000000
max	29.000000	16081.000000	10.100000	19800.000000	3345.000000

In [69]:

```
# Write your code here to generate the following output
df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29 entries, 0 to 28
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   year    29 non-null     int64
 1   roll    29 non-null     int64
 2   unem    29 non-null     float64
 3   hgrad   29 non-null     int64
 4   inc     29 non-null     int64
dtypes: float64(1), int64(4)
memory usage: 1.3 KB
```

In [70]:

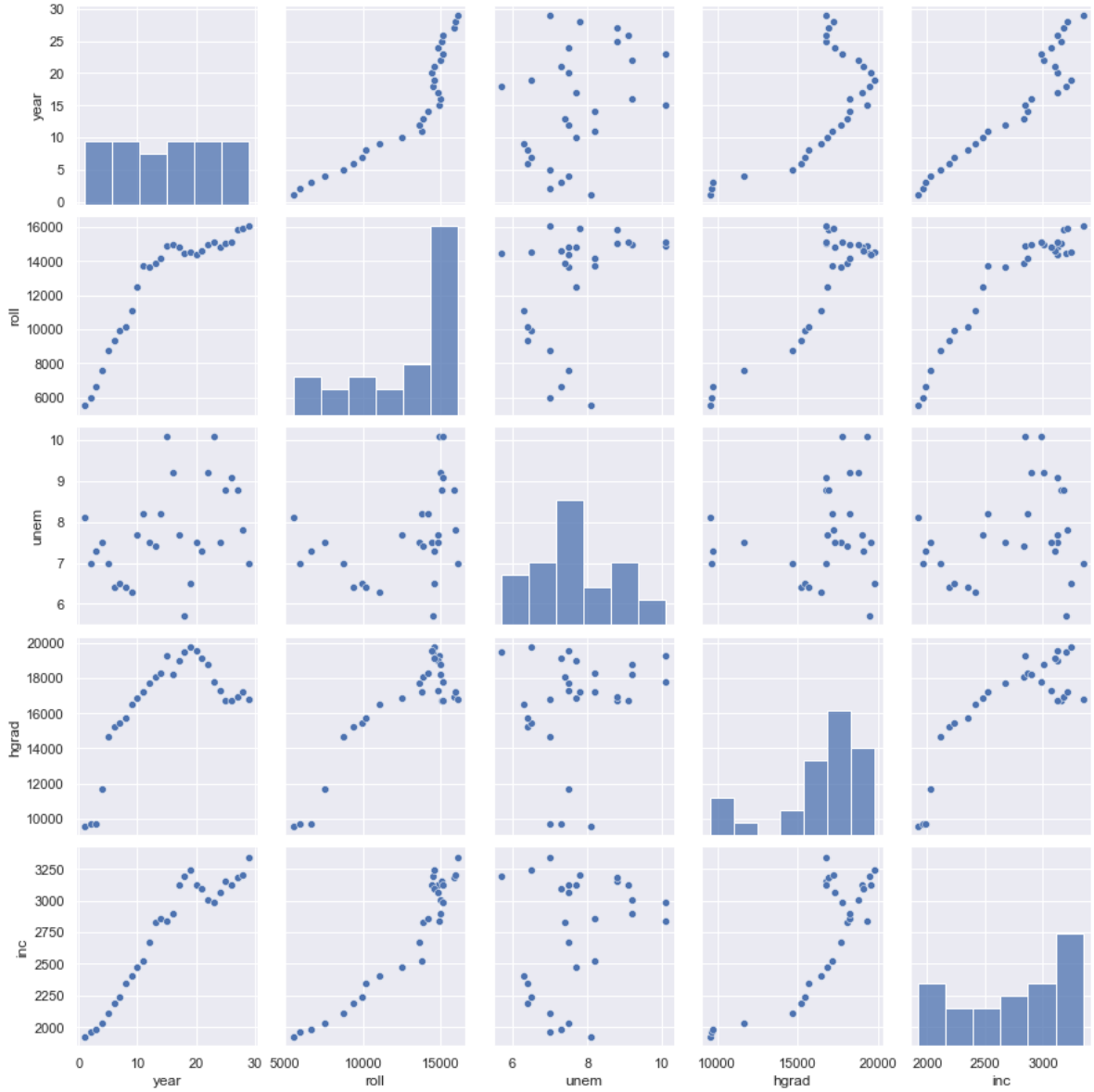
```
# Write a code for paiplot (bi-variate) analysis of the data.
```

In [71]:

```
sns.pairplot(df_data)

# seaborn.pairplot(data, hue = 'Age', diag_kind = 'kde',   plot_kws = {'edgecolor': 'k'}, s
#
```

Out[71]:

```
<seaborn.axisgrid.PairGrid at 0x188c4ed4400>
```

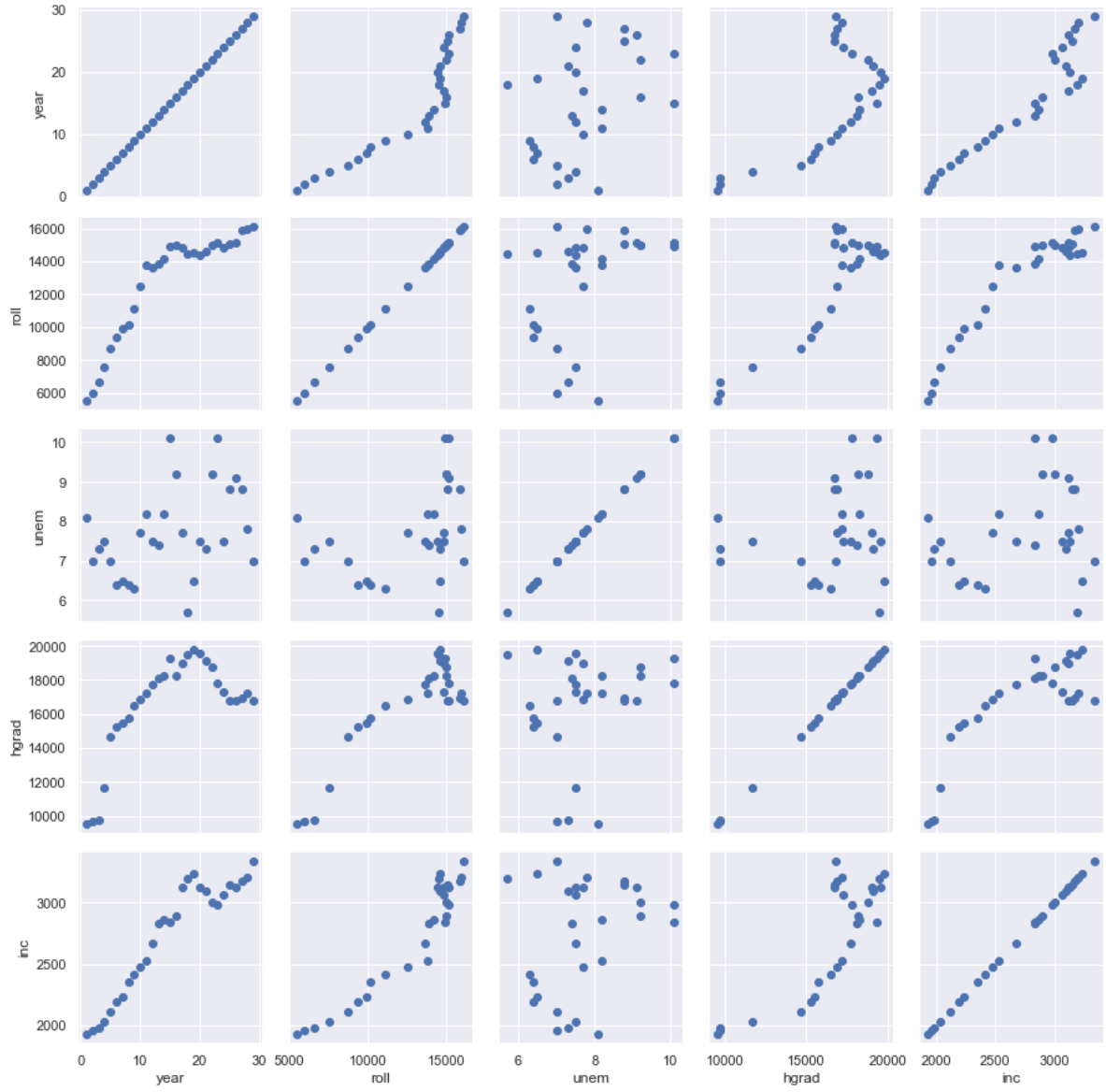


In [72]:

```
g = sns.PairGrid(df_data)
g.map(plt.scatter)
```

Out[72]:

```
<seaborn.axisgrid.PairGrid at 0x188c5b704c0>
```



Modelling

Data Prep

Use 'year', 'unem','hgrad', 'inc' as input variables and 'roll' as the output variable.

In [73]:

```
# Write your code here

X = df_data[['year', 'unem', 'hgrad','inc']]
y = df_data['roll']
```

In [74]:

```
from sklearn.model_selection import train_test_split
```

In [75]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=6)
print(X_train.shape,X_test.shape,X.shape)#Use model_selection.train_test_split from sklearn
```

```
(20, 4) (9, 4) (29, 4)
```

In [ ]:

```

```

***** Use model_selection.train_test_split from sklearn to split the data into training and testing sets. Set test_size=0.3 and random_state=6*****

In [76]:

```
# Write your code here

```

Training the Model

Now its time to train our model on our training data!

Use a multi regression model or polynomial regression model (your choice).

Fit the model using the training data.

In [83]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
```

In [84]:

```
# Write your code here

X = df_data[['year', 'unem', 'hgrad','inc']]
y = df_data['roll']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
print(X_train.shape,X_test.shape,X.shape)
```

```
(23, 4) (6, 4) (29, 4)
```

In [85]:

```
# Preparing data
from sklearn.linear_model import LinearRegression
```

In [86]:

```
lm = LinearRegression(fit_intercept=True)
```

In [87]:

```
lm.fit(X_train,y_train)
```

Out[87]:

```
LinearRegression()
```

In [88]:

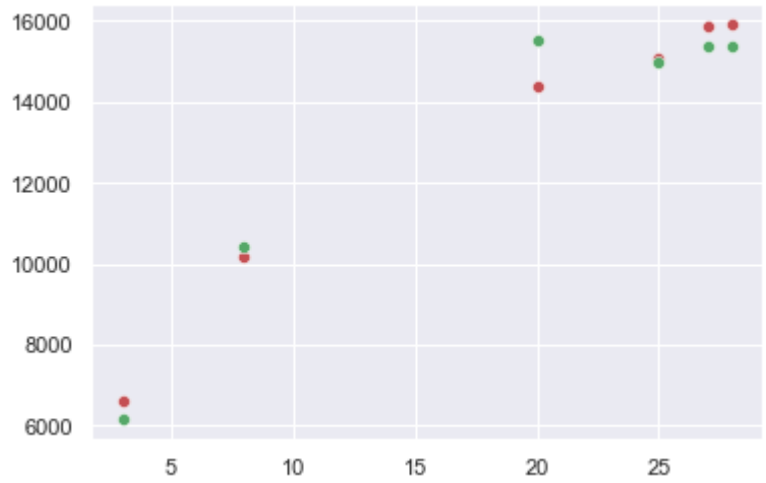
```
X = df_data[['year', 'unem', 'hgrad','inc']]
y = df_data['roll']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
print(X_train.shape,X_test.shape,X.shape)
```

```
(23, 4) (6, 4) (29, 4)
```

In [104]:

```
## builiding model
poly_model = make_pipeline(PolynomialFeatures(degree=1,include_bias=True),   LinearRegressi
## training
# print(type(X_train))
X_train = np.asarray(X_train)
y_train = np.asarray(y_train)
X_train = np.reshape(X_train,(23,4))

X_test = np.asarray(X_test)
y_test = np.asarray(y_test)
X_test = np.reshape(X_test,(6,4))

poly_model.fit(X_train, y_train)



# evaluation
yfit = poly_model.predict(X_test)
yfit_training = poly_model.predict(X_train)

sns.scatterplot(X_test[:,0],y_test,color='r') # ground_truth
sns.scatterplot(X_test[:,0],yfit,color='g') # prediction

print('RMSE Test:', np.sqrt(metrics.mean_squared_error(y_test, yfit)))
print('RMSE Training:', np.sqrt(metrics.mean_squared_error(y_train, yfit_training)))
```

```
RMSE Test: 591.026021771692
RMSE Training: 541.701409910749
```

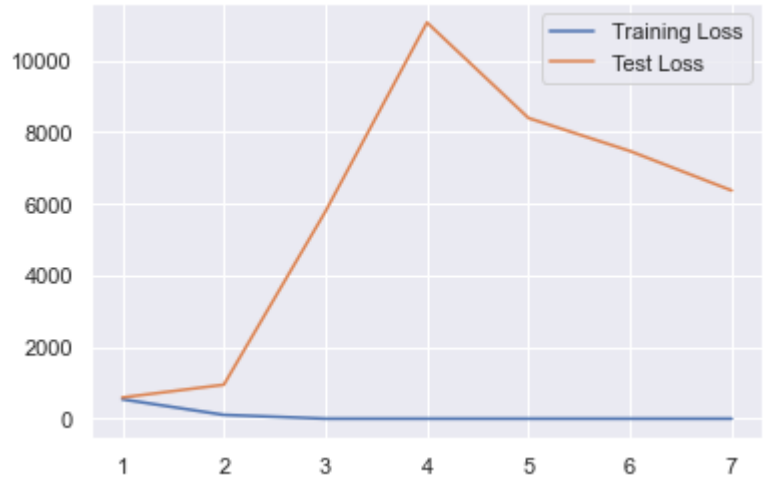


In [94]:

```
# Finding the best M
collect_training_loss = []
collect_test_loss = []
collect_m = []
for m in [1,2,3,4,5,6,7]:
    poly_model = make_pipeline(PolynomialFeatures(degree=m,include_bias=True),LinearRegress
    poly_model.fit(X_train, y_train)
    # evaluation
    yfit = poly_model.predict(X_test)
    yfit_training = poly_model.predict(X_train)

    collect_test_loss.append(np.sqrt(metrics.mean_squared_error(y_test, yfit)))
    collect_training_loss.append(np.sqrt(metrics.mean_squared_error(y_train, yfit_training)
    collect_m.append(m)

sns.lineplot(collect_m,collect_training_loss)
sns.lineplot(collect_m,collect_test_loss)
plt.legend(labels=['Training Loss', 'Test Loss'])
```

Out[94]:

```
<matplotlib.legend.Legend at 0x188c87f2d00>
```



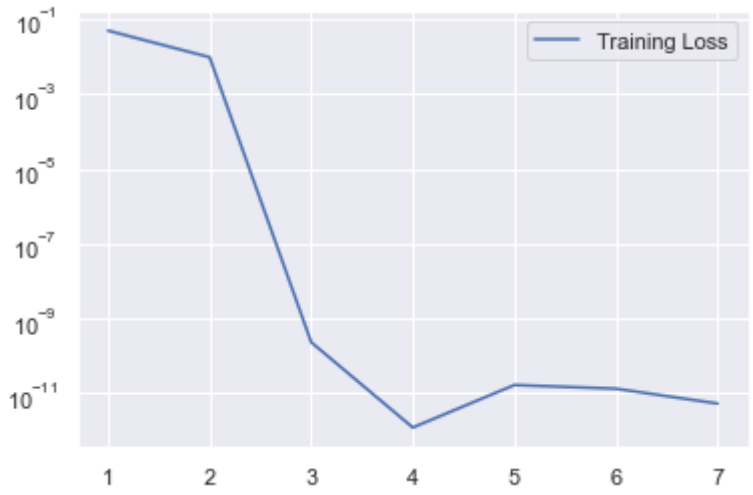
In [95]:

```
maxi = np.max([np.max(collect_training_loss),np.max(collect_test_loss)])
print(maxi)
g_results = sns.lineplot(collect_m,collect_training_loss/maxi)
g_results.set(yscale='log')

plt.legend(labels=['Training Loss'])
```

```
11059.084313452408
```

Out[95]:

```
<matplotlib.legend.Legend at 0x188c88bdca0>
```

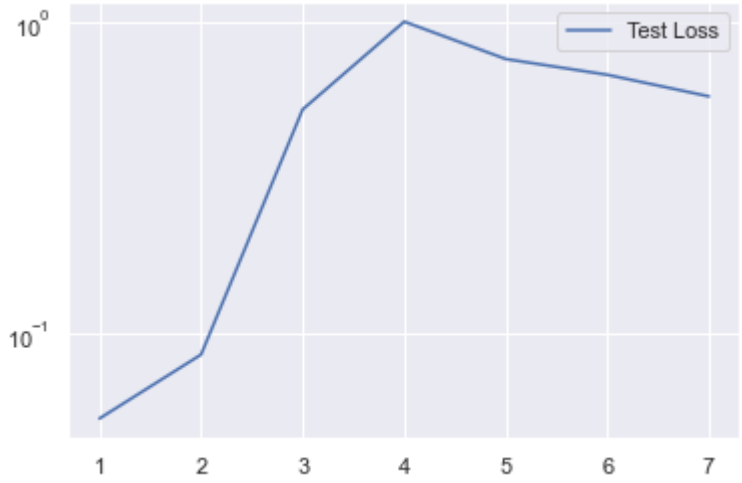


In [92]:

```
g_results = sns.lineplot(collect_m,collect_test_loss/maxi)
g_results.set(yscale='log')
plt.legend(labels=['Test Loss'])
```

Out[92]:

```
<matplotlib.legend.Legend at 0x188c8612700>
```



Predicting Test Data

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

In [96]:

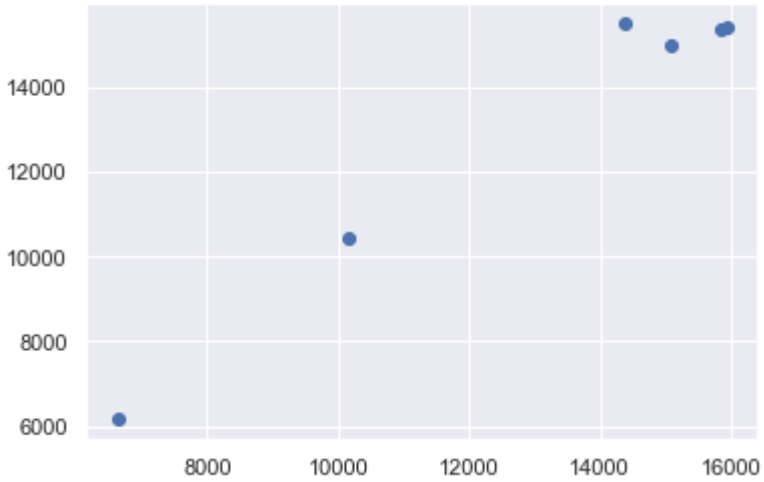
```
# Write your code here
predictions = lm.predict(X_test)
```

In [97]:

```
plt.scatter(y_test,predictions)
```

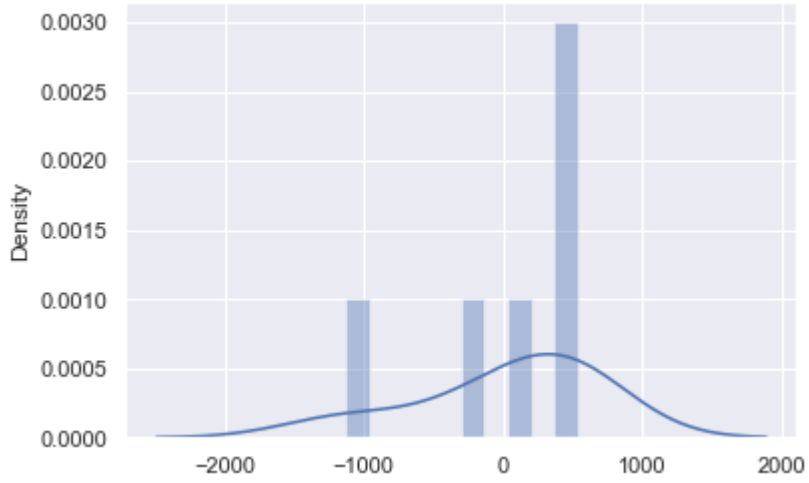
Out[97]:

```
<matplotlib.collections.PathCollection at 0x188c89b0e80>
```



In [100]:

```
sns.distplot((y_test-predictions),bins=10);
```



In [99]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 493.9191695128002
MSE: 349311.75841136946
RMSE: 591.026021771774
```

Evaluating the Model

Calculate the MAE, MSE, RMSE erros of the model on the test data.

In [32]:

```
# Write your code here
from sklearn import metrics
```

In [33]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 557.7575371626649
MSE: 369520.63219932216
RMSE: 607.8820874144279
```

In [ ]:

```

```