In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split, LeaveOneOut, KFold, cross_val_score
from sklearn.metrics import mean_squared_error
import sklearn.linear_model as lm
import statsmodels.formula.api as smf
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remoun
t=True).
```

In [ ]:
```
Auto = pd.read_csv('/content/gdrive/MyDrive/ISLR/Auto.csv',na_values='?').dropna()
Auto.shape
```

Out[27]:
```
(392, 9)
```

In [ ]:
```
np.random.seed(1)
##training model
train = np.random.choice(Auto.shape[0],196,replace=False)
select = np.in1d(range(Auto.shape[0]),train)
lm1 = smf.ols('mpg~horsepower',data=Auto[select]).fit()
print(lm1.summary().tables[1])
lm2 = smf.ols('mpg~horsepower+I(horsepower**2)',data=Auto[select]).fit()
lm2.summary().tables[1]
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     40.3338      1.023     39.416      0.000      38.316      42.352
horsepower    -0.1596      0.009    -17.788      0.000      -0.177      -0.142
==============================================================================
```

Out[32]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	60.3022	2.562	23.541	0.000	55.250	65.354
horsepower	-0.5186	0.044	-11.807	0.000	-0.605	-0.432
I(horsepower ** 2)	0.0014	0.000	8.302	0.000	0.001	0.002

In [ ]:
```
```

In [ ]:
```
Auto.head(2)
```

Out[23]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320

In [ ]:
```
#testing
pred = lm1.predict(Auto)
error = (Auto['mpg']-pred)**2
print(np.mean(error[~select]))  ##error of test data set
```

```
23.361902892587224
```

In [ ]:

In [ ]:
```
lm2 = smf.ols('mpg ~ horsepower+I(horsepower**2)',data=Auto[select]).fit()
prd1=lm2.predict(Auto)
error1 = (Auto['mpg']-prd1)**2
print(np.mean(error1[~select]))
```

```
20.25269085835004
```

In [ ]:
```
lm3 = smf.ols('mpg ~ horsepower+I(horsepower**2)+I(horsepower**3)',data=Auto[select]).fit()
prd2=lm3.predict(Auto)
error2 = (Auto['mpg']-prd2)**2
print(np.mean(error2[~select]))
```

```
20.32560936577359
```

In [ ]:
```
l1 = smf.ols('mpg~horsepower',data=Auto).fit()
l1.summary().tables[1]
```

Out[17]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	39.9359	0.717	55.660	0.000	38.525	41.347
horsepower	-0.1578	0.006	-24.489	0.000	-0.171	-0.145

In [ ]:
```
Auto["HPS"]=Auto.horsepower**2
Auto.head(2)
Auto[["horsepower","HPS"]]
```

Out[15]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	HPS
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	16900.0
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	27225.0

In [ ]:
```
x = pd.DataFrame(Auto.horsepower)
x1 =Auto[["horsepower","HPS"]]
y = Auto.mpg
```

In [ ]:
```
l1 = smf.ols('mpg~horsepower',data=Auto).fit()
l1.summary().tables[1]
```

Out[17]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	39.9359	0.717	55.660	0.000	38.525	41.347
horsepower	-0.1578	0.006	-24.489	0.000	-0.171	-0.145

In [ ]:
```
model = lm.LinearRegression()
model.fit(x,y)
print(model.coef_)
print(model.intercept_)
```

```
[-0.15784473]
39.93586102117047
```

In [ ]:
```
model2 = lm.LinearRegression()
model2.fit(x1,y)
print(model2.coef_)
print(model2.intercept_)
```

```
[-0.46618963  0.00123054]
56.90009970211294
```

In [ ]:
```
import sklearn.linear_model as lm
x = pd.DataFrame(Auto.horsepower)
y = Auto.mpg
model = lm.LinearRegression()
model.fit(x,y)
print(model.coef_)
print(model.intercept_)
```

```
[-0.15784473]
39.93586102117047
```

In [ ]:
```
x.shape[0]
```

Out[28]:
```
392
```

In [ ]:
```
##LOCV
kflod = KFold(n_splits=x.shape[0])
test = cross_val_score(model,x,y,cv = kflod,scoring='neg_mean_squared_error',n_jobs=-1)
print(np.mean(-test))
```

```
19.24821312448967
```

In [ ]:
```
kflod


 = KFold(n_splits=10,random_state=1, shuffle=True)
test5 = cross_val_score(model,x,y,cv = kflod5,scoring='neg_mean_squared_error',n_jobs=-1)
test5
print(np.mean(-test5))
```

Out[7]:
```
array([-25.65015985, -30.29781332, -23.56550989, -20.75383058,
       -23.42372298, -32.93506361, -21.89071342, -27.35148938,
       -17.85005823, -17.25839605])
```

In [ ]:

```
#1. check NA values. if NA value then replace with mean,median or mode
#2. convert X's(only quatitative variable) and Y to scale( for this sem convert on X's)
##3. Check VIF
###3.a for qualtitative data set do onehot encoding
##4. Split the data set(train, test(80%,20%) or K_fold)
##5. check non linear realtionship
##6. findout the best traning models where R2, adj R2 is high, all p_values sig, AIC,BIC low
##6.i. check high leverage point
##7. findout the best testing models from MSEs.
##8. then pick the model which is good in traning and testing
###6.a. findout the best modelsfrom whole dataset.
###7.a. then go kflod CV to find out the besting testing model through MSEs.
##8.a. then pick the model which is good in traning and testing
```

In [ ]:

```
```

In [ ]:

```
poly = PolynomialFeatures(2)
X_poly=poly.fit_transform(Auto.horsepower.values.reshape(-1,1))
X_train, X_test, y_train, y_test = train_test_split(X_poly, Auto.mpg.ravel(), test_size=0.5, random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(196, 3)
(196, 3)
(196,)
(196,)
```

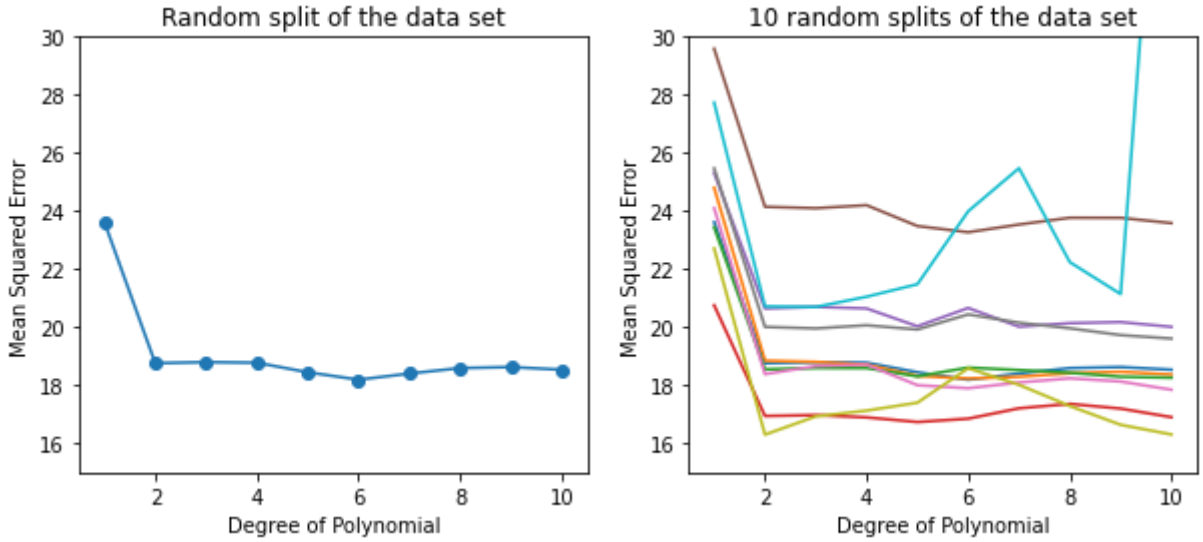
In [ ]:

```
lr = lm.LinearRegression()
lr.fit(X_train, y_train)
pred = lr.predict(X_test)
Z= mean_squared_error(y_test, pred)
print(Z)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-22fa71d579d7> in <module>
----> 1 lr = lm.LinearRegression()
      2 lr.fit(X_train, y_train)
      3 pred = lr.predict(X_test)
      4 Z= mean_squared_error(y_test, pred)
      5 print(Z)

/usr/local/lib/python3.8/dist-packages/statsmodels/base/wrapper.py in __getattribute__(self, attr)
     32             pass
     33
---> 34         obj = getattr(results, attr)
     35         data = results.model.data
     36         how = self._wrap_attrs.get(attr)

AttributeError: 'OLSResults' object has no attribute 'LinearRegression'
```

In [ ]:

```
##Cross Validation
t_prop = 0.5
p_order = np.arange(1,11)
r_state = np.arange(0,10)


, Y = np.meshgrid(p_order, r_state, indexing='ij')
Z = np.zeros((p_order.size,r_state.size))

regr = lm.LinearRegression()

# Generate 10 random splits of the dataset
for (i,j),v in np.ndenumerate(Z):
    poly = PolynomialFeatures(int(X[i,j]))
    X_poly = poly.fit_transform(Auto.horsepower.values.reshape(-1,1))

    X_train, X_test, y_train, y_test = train_test_split(X_poly, Auto.mpg.ravel(),
                                                        test_size=t_prop, random_state=Y[i,j])

    regr.fit(X_train, y_train)
    pred = regr.predict(X_test)
    Z[i,j]= mean_squared_error(y_test, pred)

fig, (ax1, ax2) = plt.subplots(1,2, figsize=(10,4))

# Left plot (first split)
ax1.plot(X.T[0],Z.T[0], '-o')
ax1.set_title('Random split of the data set')

# Right plot (all splits)
ax2.plot(X,Z)
ax2.set_title('10 random splits of the data set')

for ax in fig.axes:
    ax.set_ylabel('Mean Squared Error')
    ax.set_ylim(15,30)
    ax.set_xlabel('Degree of Polynomial')
    ax.set_xlim(0.5,10.5)
    ax.set_xticks(range(2,11,2));
```



In [ ]:
```
X = Auto.iloc[:, 1:8]

#output
Y = Auto.iloc[:, 0]
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
lreg = LinearRegression()
lreg.fit(X, Y)
print(lreg.coef_)
```

```
[-0.49337632  0.01989564 -0.01695114 -0.00647404  0.08057584  0.75077268
  1.4261405 ]
```

In [ ]:
```
Auto.head(2)
```

Out[4]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320

In [ ]:
```
from sklearn.linear_model import Ridge
ridgeR = Ridge(alpha = 280)
ridgeR.fit(X, Y)
ridgeR.coef_
```

Out[17]:
```
array([-0.1080397 ,  0.00693077, -0.00879861, -0.00660617,  0.0650845 ,
        0.7061361 ,  0.47513038])
```

In [ ]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha = 1)
lasso.fit(X,Y)
lasso.coef_
```

Out[18]:
```
array([-0.        ,  0.        , -0.00734394, -0.00646937,  0.        ,
        0.66308442,  0.        ])
```

In [ ]:
```

```