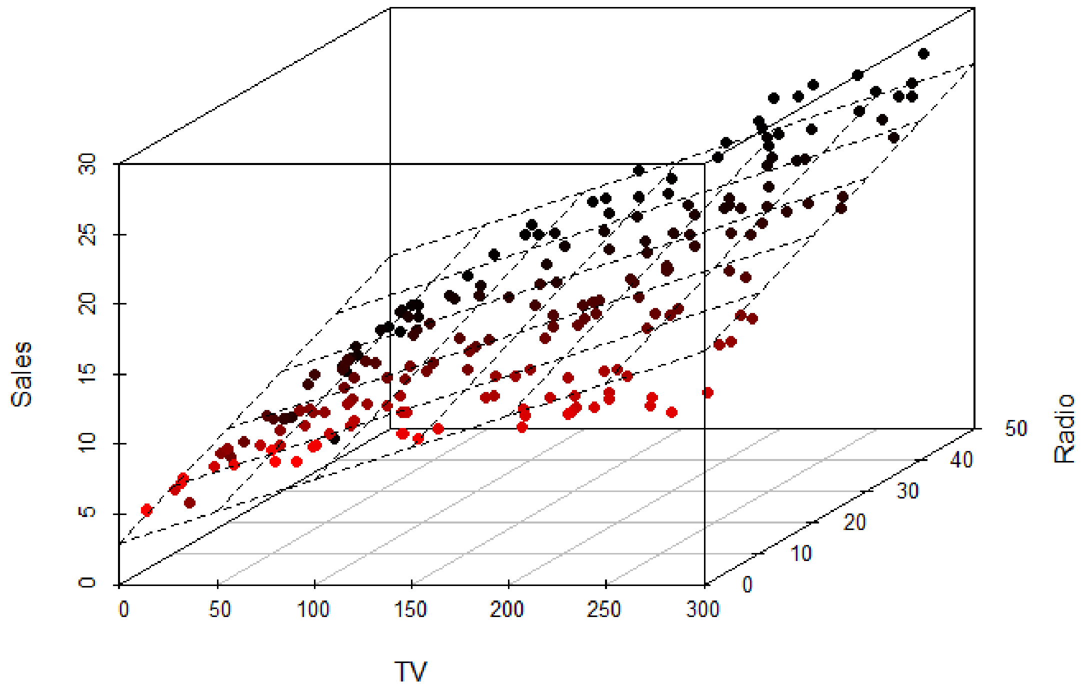
Multiple Linear Regression For Prediction



Source: Python plot posted to StackExchange https://stackoverflow.com/questions/26431800/plot-linear-model-in-3d-with-matplotlib

Explanatory vs. Predictive

Explain/describe population relationships

- Small sample, few variables
- Retrospective
- Find good fitting regression model
- Confidence intervals, hypothesis test, p-value

Predict values of new records

- Large sample, many variables
- Prospective
- Regression with high predictive power
- Predictive power on holdout data

Example – Used Toyota Car Prices

Scenario – Toyota dealers accept used cars in trade-ins, need predictive model to know how much to offer the customer

The Regression Equation

outcome

predictors

Error
(noise)

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

constant

coefficients

The Regression Equation

Training phase – known outcome

predictors

Error (noise)

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

constant

coefficients

The Regression Equation

Scoring phase - unknown outcome

predictors

Error (noise)

$$? = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

Constant (estimated in the training phase)

Coefficients (estimated in the training phase)

Used Toyota Car Prices – Predictors

Variable	Description
Price	Offer price in Euros
Age	Age in months as of August 2004
Kilometers	Accumulated kilometers on odometer
Fuel Type	Fuel type (*Petrol*, *Diesel*, *CNG*)
HP	Horsepower
Metallic	Metallic color? (Yes = 1, No = 0)
Automatic	Automatic (Yes = 1, No = 0)
CC	Cylinder volume in cubic centimeters
Doors	Number of doors
QuartTax	Quarterly road tax in Euros
Weight	Weight in kilograms

Import Needed Functionality

```
import itertools
import pandas as pd
import statsmodels.formula.api as sm
from sklearn.feature_selection import
     SelectKBest, f_regression
from utilities import regressionSummary
```

Prepare and Partition Data

```
# Reduce data frame to the top 1000 rows and select
# columns for regression analysis
car_df = pd.read_csv(DATA / 'ToyotaCorolla.csv’,
    encoding='latin-1')
selected_var = [car_df.columns[i] for i in (2, 3, 6,
    7, 8, 9, 11, 12, 13, 16, 17)]
car_df = car_df.iloc[0:1000]

# Partition data
train_df = car_df.sample(frac=0.6, random_state=1)
valid_df = car_df.drop(train_df.index)
```

Prepare and Partition Data

```
# Reduce data frame to the top 1000 rows and select
# columns for regression analysis
car_df = pd.read_csv(DATA / 'ToyotaCorolla.csv’,
    encoding='latin-1')
selected_var = [car_df.columns[i] for i in (2, 3, 6,
    7, 8, 9, 11, 12, 13, 16, 17)]
car_df = car_df.iloc[0:1000]

# Partition data
train_df = car_df.sample(frac=0.6, random_state=1)
valid_df = car_df.drop(train_df.index)
```

Prepare and Partition Data

```
# Reduce data frame to the top 1000 rows and select
# columns for regression analysis
car_df = pd.read_csv(DATA / 'ToyotaCorolla.csv',
    encoding='latin-1')
selected_var = [car_df.columns[i] for i in (2, 3, 6,
    7, 8, 9, 11, 12, 13, 16, 17)]
car_df = car_df.iloc[0:1000]

# Partition data
train_df = car_df.sample(frac=0.6, random_state=1)
valid_df = car_df.drop(train_df.index)
```

Prepare and Partition Data

```
# Reduce data frame to the top 1000 rows and select
# columns for regression analysis
car_df = pd.read_csv(DATA / 'ToyotaCorolla.csv',
    encoding='latin-1')
selected_var = [car_df.columns[i] for i in (2, 3, 6,
    7, 8, 9, 11, 12, 13, 16, 17)]
car_df = car_df.iloc[0:1000]

# Partition data
train_df = car_df.sample(frac=0.6, random_state=1)
valid_df = car_df.drop(train_df.index)
```

Prepare and Partition Data

```
# Reduce data frame to the top 1000 rows and select
# columns for regression analysis
car_df = pd.read_csv(DATA / 'ToyotaCorolla.csv',
    encoding='latin-1')
selected_var = [car_df.columns[i] for i in (2, 3, 6,
    7, 8, 9, 11, 12, 13, 16, 17)]
car_df = car_df.iloc[0:1000]

# Partition data
train_df = car_df.sample(frac=0.6, random_state=1)
valid_df = car_df.drop(train_df.index)
```

Fit the regression

```
# Construct regression formula and use sm.ols to run a
# linear regression of Price
# on the remaining 11 predictors in the training set

independent_var = list(selected_var)
independent_var.remove('Price')
formula = 'Price ~ ' + ' + '.join(independent_var)
car_lm = sm.ols(formula=formula, data=train_df).fit()
car_lm.summary()
```

Fit the regression

```
# Construct regression formula and use sm.ols to run a
# linear regression of Price
# on the remaining 11 predictors in the training set

independent_var = list(selected_var)
independent_var.remove('Price')
formula = 'Price ~ ' + ' + '.join(independent_var)
car_lm = sm.ols(formula=formula, data=train_df).fit()
car_lm.summary()
```

Review Results

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.867
Model:	OLS	Adj. R-squared:	0.864
Method:	Least Squares	F-statistic:	347.4
Date:	Mon, 18 Jun 2018	Prob (F-statistic):	7.53e-249
Time:	19:44:28	Log-Likelihood:	-5183.5
No. Observations:	600	AIC:	1.039e+04
Df Residuals:	588	BIC:	1.044e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-6314.1254	1822.923	-3.464	0.001	-9894.358	-2733.893
Age_08_04	-127.4934	4.896	-26.038	0.000	-137.110	-117.877
KM	-0.0206	0.002	-8.929	0.000	-0.025	-0.016
HP	36.8883	5.115	7.212	0.000	26.843	46.933
Met_Color	82.3558	123.302	0.668	0.504	-159.810	324.521
Automatic	143.5263	289.166	0.496	0.620	-424.397	711.449
CC	0.0236	0.097	0.243	0.808	-0.167	0.215
Doors	-46.6678	62.884	-0.742	0.458	-170.172	76.836
Quarterly_Tax	12.1313	2.851	4.255	0.000	6.532	17.730
Weight	18.1393	1.822	9.954	0.000	14.560	21.718
Fuel_Type_Diesel	-31.4359	569.873	-0.055	0.956	-1150.670	1087.799
Fuel_Type_Petrol	1461.2586	574.021	2.546	0.011	333.877	2588.640

Omnibus:	112.122	Durbin-Watson:	1.946
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1305.165
Skew:	-0.420	Prob(JB):	3.86e-284
Kurtosis:	10.176	Cond. No.	2.34e+06

Review Results

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.867
Model:	OLS	Adj. R-squared:	0.864
Method:	Least Squares	F-statistic:	347.4
Date:	Mon, 18 Jun 2018	Prob (F-statistic):	7.53e-249
Time:	19:44:28	Log-Likelihood:	-5183.5
No. Observations:	600	AIC:	1.039e+04
Df Residuals:	588	BIC:	1.044e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-6314.1254	1822.923	-3.464	0.001	-9894.358	-2733.893
Age_08_04	-127.4934	4.896	-26.038	0.000	-137.110	-117.877
KM	-0.0206	0.002	-8.929	0.000	-0.025	-0.016
HP	36.8883	5.115	7.212	0.000	26.843	46.933
Met_Color	82.3558	123.302	0.668	0.504	-159.810	324.521
Automatic	143.5263	289.166	0.496	0.620	-424.397	711.449
CC	0.0236	0.097	0.243	0.808	-0.167	0.215
Doors	-46.6678	62.884	-0.742	0.458	-170.172	76.836
Quarterly_Tax	12.1313	2.851	4.255	0.000	6.532	17.730
Weight	18.1393	1.822	9.954	0.000	14.560	21.718
Fuel_Type_Diesel	-31.4359	569.873	-0.055	0.956	-1150.670	1087.799
Fuel_Type_Petrol	1461.2586	574.021	2.546	0.011	333.877	2588.640

Omnibus:	112.122	Durbin-Watson:	1.946
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1305.165
Skew:	-0.420	Prob(JB):	3.86e-284
Kurtosis:	10.176	Cond. No.	2.34e+06

Score the Validation Data

```
# Use predict() to make predictions on a new set

car_lm_pred = car_lm.predict(valid_df)
result = pd.DataFrame({'Predicted': car_lm_pred,
     'Actual': valid_df.Price,'Residual’:
     valid_df.Price - car_lm_pred })
```

Look at First 10 Rows - Validation

Output

```
Actual        Predicted           Residual
1   13750    16206.480671    -2456.480671
7   18600    16704.259643     1895.740357
10  20950    21003.848119      -53.848119
15  22000    20883.866334     1116.133666
20  15950    15039.157810      910.842190
21  16950    17078.992106     -128.992106
22  15950    15799.618511      150.381489
24  16250    16302.187336      -52.187336
25  15950    16757.750599     -807.750599
26  17495    16377.003081     1117.996919
```

Mean Error (ME)

Output

Actual		Predicted	Residual
1	13750	16206.480671	-2456.480671
7	18600	16704.259643	1895.740357
10	20950	21003.848119	-53.848119
15	22000	20883.866334	1116.133666
20	15950	15039.157810	910.842190
21	16950	17078.992106	-128.992106
22	15950	15799.618511	150.381489
24	16250	16302.187336	-52.187336
25	15950	16757.750599	-807.750599
26	17495	16377.003081	1117.996919

Average of the residuals

Mean Error

Output

Actual		Predicted	Residual
1	13750	16206.480671	-2456.480671
7	18600	16704.259643	1895.740357
10	20950	21003.848119	-53.848119
15	22000	20883.866334	1116.133666
20	15950	15039.157810	910.842190
21	16950	17078.992106	-128.992106
22	15950	15799.618511	150.381489
24	16250	16302.187336	-52.187336
25	15950	16757.750599	-807.750599
26	17495	16377.003081	1117.996919

Average
$280

Root Mean Squared Error (RMSE)

Output

Actual		Predicted	Residual
1	13750	16206.480671	-2456.480671
7	18600	16704.259643	1895.740357
10	20950	21003.848119	-53.848119
15	22000	20883.866334	1116.133666
20	15950	15039.157810	910.842190
21	16950	17078.992106	-128.992106
22	15950	15799.618511	150.381489
24	16250	16302.187336	-52.187336
25	15950	16757.750599	-807.750599
26	17495	16377.003081	1117.996919

Square the residuals, average them, take square root

Mean Absolute Error (MAE)

Output

	Actual	Predicted	Residual
1	13750	16206.480671	-2456.480671
7	18600	16704.259643	1895.740357
10	20950	21003.848119	-53.848119
15	22000	20883.866334	1116.133666
20	15950	15039.157810	910.842190
21	16950	17078.992106	-128.992106
22	15950	15799.618511	150.381489
24	16250	16302.187336	-52.187336
25	15950	16757.750599	-807.750599
26	17495	16377.003081	1117.996919

Take absolute values of residuals, find average

Mean Absolute Percentage Error (MAPE)

Output

	Actual	Predicted	Residual
1	13750	16206.480671	-2456.480671
7	18600	16704.259643	1895.740357
10	20950	21003.848119	-53.848119
15	22000	20883.866334	1116.133666
20	15950	15039.157810	910.842190
21	16950	17078.992106	-128.992106
22	15950	15799.618511	150.381489
24	16250	16302.187336	-52.187336
25	15950	16757.750599	-807.750599
26	17495	16377.003081	1117.996919

Take residual as % of actual, find absolute value, find average

Review Error Metrics - Validation

```
              Regression statistics

                   Mean Error (ME)   : 49.1865
     Root Mean Squared Error (RMSE) : 1349.2010
          Mean Absolute Error (MAE) : 1021.6062
        Mean Percentage Error (MPE) : 0.0381
Mean Absolute Percentage Error (MAPE) : 9.0735
```

Used Toyota Car Prices – Predictors

Variable	Description
Price	Offer price in Euros
Age	Age in months as of August 2004
Kilometers	Accumulated kilometers on odometer
Fuel Type	Fuel type (*Petrol*, *Diesel*, *CNG*)
HP	Horsepower
Metallic	Metallic color? (Yes = 1, No = 0)
Automatic	Automatic (Yes = 1, No = 0)
CC	Cylinder volume in cubic centimeters
Doors	Number of doors
QuartTax	Quarterly road tax in Euros
Weight	Weight in kilograms