Supervised Learning

- Supervised Learning
  - 1. Examples of Supervised Learning
    - 1.1. Weather Apps
    - 1.2. Email Filters
    - 1.3. Real Estate
    - 1.4. Supervised Learning Case Study
  - 2. Understanding the Algorithm behind Supervised Learning
    - 2.1. Supervised Learning Flow
    - 2.2. Training and Testing
    - 2.3. Prediction based on Training and Testing
  - 3. Types of Supervised Learning
    - 3.1. Classification Algorithm
    - 3.2. Regression Algorithm
    - 3.3. Types of Classification Algorithms
    - 3.4. Types of Regression Algorithms
    - 3.5. Linear Regression
    - 3.6. Multiple linear regression
    - 3.7. Polynomial regression
    - 3.8. Regression Use Case
  - 4. Accuracy Metrics
    - 4.1. Accuracy Metrics
    - 4.2. Adjusted R Squared
    - 4.3. Gradient Descent
    - 4.4. Gradient Descent
    - 4.5. Evaluating Coefficients
  - 5. Case Study
    - 5.1. Step 1: Decide on the relationship between the features and the target variable using scatter plots
    - 5.2. Step 2: Check the correlation between the features and the target variable using pairplot
    - 5.3. Step 3: Check the correlation between the features using heatmap
    - 5.4. Step 4: Check the correlation between the features and the target variable using scatter plots
    - 5.5. Step 5: Regressional plots and jointplots
    - 5.6. Step 6: Handling categorical predictors with two categories
    - 5.7. Step 7: Dummy variable creation
    - 5.8. Step 8: Feature Selection
    - 5.9. Step 9: Feature Selection
    - 5.10. Step 10: Handling categorical predictors with two categories
    - 5.11. Step 11: Creating a training set and a test set
    - 5.12. Step 12: Scaling the training set
    - 5.13. Step 13: Feature Selection
    - 5.14. Step 14: Training the linear regression model
    - 5.15. Step 15: Checking the coefficients
    - 5.16. Step 16: Checking Bias
    - 5.17. Step 17: Checking Variance
    - 5.18. Step 18: Checking the residuals
    - 5.19. Step 19: Checking the residuals
    - 5.20. Step 20: Checking the residuals
    - 5.21. Step 21: Predictions on the test set
    - 5.22. Step 22: Rescaling the coefficients
    - 5.23. Step 23: Interpretation
    - 5.24. Step 24: Residual analysis
    - 5.25. Step 25: Residual analysis
    - 5.26. Step 26: Residual analysis
    - 5.27. Step 27: Residual analysis
    - 5.28. Step 28: Residual analysis
    - 5.29. Step 29: Residual analysis
    - 5.30. Step 30: Residual analysis
    - 5.31. Step 31: Residual analysis
    - 5.32. Step 32: Residual analysis
    - 5.33. Step 33: Residual analysis
    - 5.34. Step 34: Residual analysis
    - 5.35. Step 35: Residual analysis
    - 5.36. Step 36: Residual analysis
    - 5.37. Step 37: Residual analysis
    - 5.38. Step 38: Residual analysis
    - 5.39. Step 39: Residual analysis
    - 5.40. Step 40: Residual analysis
    - 5.41. Step 41: Residual analysis
    - 5.42. Step 42: Residual analysis
    - 5.43. Step 43: Residual analysis
    - 5.44. Step 44: Residual analysis
    - 5.45. Step 45: Residual analysis
    - 5.46. Step 46: Residual analysis
    - 5.47. Step 47: Residual analysis
    - 5.48. Step 48: Residual analysis
    - 5.49. Step 49: Residual analysis
    - 5.50. Step 50: Residual analysis
    - 5.51. Step 51: Residual analysis
    - 5.52. Step 52: Residual analysis
    - 5.53. Step 53: Residual analysis
    - 5.54. Step 54: Residual analysis
    - 5.55. Step 55: Residual analysis
    - 5.56. Step 56: Residual analysis
    - 5.57. Step 57: Residual analysis
    - 5.58. Step 58: Residual analysis
    - 5.59. Step 59: Residual analysis
    - 5.60. Step 60: Residual analysis
    - 5.61. Step 61: Residual analysis
    - 5.62. Step 62: Residual analysis
    - 5.63. Step 63: Residual analysis
    - 5.64. Step 64: Residual analysis
    - 5.65. Step 65: Residual analysis
    - 5.66. Step 66: Residual analysis
    - 5.67. Step 67: Residual analysis
    - 5.68. Step 68: Residual analysis
    - 5.69. Step 69: Residual analysis
    - 5.70. Step 70: Residual analysis
    - 5.71. Step 71: Residual analysis
    - 5.72. Step 72: Residual analysis
    - 5.73. Step 73: Residual analysis
    - 5.74. Step 74: Residual analysis
    - 5.75. Step 75: Residual analysis
    - 5.76. Step 76: Residual analysis
    - 5.77. Step 77: Residual analysis
    - 5.78. Step 78: Residual analysis
    - 5.79. Step 79: Residual analysis
    - 5.80. Step 80: Residual analysis
    - 5.81. Step 81: Residual analysis
    - 5.82. Step 82: Residual analysis
    - 5.83. Step 83: Residual analysis
    - 5.84. Step 84: Residual analysis
    - 5.85. Step 85: Residual analysis
    - 5.86. Step 86: Residual analysis
    - 5.87. Step 87: Residual analysis
    - 5.88. Step 88: Residual analysis
    - 5.89. Step 89: Residual analysis
    - 5.90. Step 90: Residual analysis
    - 5.91. Step 91: Residual analysis
    - 5.92. Step 92: Residual analysis
    - 5.93. Step 93: Residual analysis
    - 5.94. Step 94: Residual analysis
    - 5.95. Step 95: Residual analysis
    - 5.96. Step 96: Residual analysis
    - 5.97. Step 97: Residual analysis
    - 5.98. Step 98: Residual analysis
    - 5.99. Step 99: Residual analysis
    - 5.100. Step 100: Residual analysis
    - 5.101. Step 101: Residual analysis
    - 5.102. Step 102: Residual analysis
    - 5.103. Step 103: Residual analysis
    - 5.104. Step 104: Residual analysis
    - 5.105. Step

```
>>> ggplot(data=df,aes(x='Newspapers',y='sales')) +\
...     geom_point()
```

We do see a relationship between sales and radio. Applying linear regression analysis to estimate sales and television or spending.

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression()
>>> lr.fit(df[['TV']],df[['sales']])
LinearRegression(copy_X=True, fit_intercept=True, normalize=False)
```

Step 2: Apply Linear Regression

```
>>> lr.coef_
array([[ 0.04753664]])
>>> lr.intercept_
array([ 7.03259355])
>>> lr.score(df[['TV']],df[['sales']])
0.611875050850071
```

Let's say that there was a new market where TV advertising spend were $50,000. What would our predicts the sales in that market?

Use the previous equation to calculate the sales.

```
>>> 7.03259355 + 0.04753664*50
9.40942557
```

and predict the new sales

```
>>> lr.predict(50)
Traceback (most recent call last):
  File "<stdout>", line 1, in <module>
  File "/usr/local/lib/python2.7/site-packages/numpy/core/numeric.py", line 190, in asarray
    raise TypeError("Cannot convert %s to array" % str(a))
TypeError: Cannot convert 50 to array
>>> lr.predict(50*1)
array([[ 9.40942557]])
```

Using the above equation, we would now predict the sales of 9.4940 widgets in that market.

```
>>> lr.predict(1,0)
array([[ 7.08012019]])
```

Let's make predictions for the multiple values in the dataset using the dataset sales to plot the regression line against the actual values.

```
>>> lr.predict(df[['TV']])[0:5,0]*1000
array([ 9.40942557,  7.51973818,  7.70939319,  7.89904819,  8.47837321])
>>> df[0:5][['sales']]*1000
   sales
0   9.6
1   7.6
2   9.2
3   7.9
4   8.7
```

Looking at the data you can see that our model is very close to the actual sales values.

Step 3: Hypothesis testing and p-values

Use the Ord hypothesis (let's assume there is a relationship between tv and sales. Now let's test the hypothesis, we start open the hypothesis that there is no relationship between tv and sales.

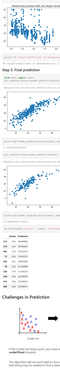
```
>>> from sklearn.feature_selection import f_regression
>>> f_regression(df[['TV']],df[['sales']])
(array([ 312.14499533]), array([ 1.46779657e-39]))
>>> zip(zip(df.columns,lr.coef_),zip(f_regression(df[['TV']],df[['sales']])[0],
... zip(f_regression(df[['TV']],df[['sales']])[1],lr.predict(df[['TV']])[0:5,0]*1000))
[(('TV', 0.04753664), (312.14499533, (1.4677965700000001e-39, 9.40942557))),
 (('Newspapers', 0.054693000000000001), (15.811088, (0.00012001000000000001, 7.51973818)),
 (('sales', 1.0000000000000000), (inf, (0.0, 7.70939319)),
 (('Radio', 0.20249999999999999), (332.11329000000001, (8.3916000000000000e-41, 7.89904819)),
 (('Intercept', 7.0325935500000000), (0.0, (0.0, 8.47837321)))
>>>
>>> df.corr()
          TV  Newspapers     sales     Radio
TV  1.000000    0.056648  0.782224  0.054809
Newspapers  0.056648    1.000000  0.228299  0.354104
sales       0.782224    0.228299  1.000000  0.576748
Radio       0.054809    0.354104  0.576748  1.000000
>>>
>>> df[['TV','Radio','Newspapers']].corrwith(df[['sales']])
TV            0.782224
Radio         0.576748
Newspapers    0.228299
dtype: float64
>>>
>>> ggplot(df[['TV','Radio','Newspapers','sales']],\
...     aes(x='TV',y='Radio',size='Newspapers',color='sales')) +\
...     geom_point()
```

Looking at the chart you can tell that most high sales are coming from tv and radio.

Step 4: Multiple Linear Regression

Use the Ord hypothesis (let's assume there is a relationship between tv and sales. Now let's test the hypothesis, we start open the hypothesis that there is no relationship between tv and sales.

```
>>> lr = LinearRegression()
>>> lr.fit(df[['TV','Radio']],df[['sales']])
LinearRegression(copy_X=True, fit_intercept=True, normalize=False)
>>> lr.coef_
array([[ 0.04575482,  0.18799423]])
>>> lr.intercept_
array([ 2.92109991])
>>> lr.score(df[['TV','Radio']],df[['sales']])
0.897194692937763
>>> f_regression(df[['TV','Radio']],df[['sales']])
(array([ 859.58275449,  548.20991694]), array([ 8.35570040e-112,  6.39467667e-086]))
>>> zip(zip(df.columns,lr.coef_),zip(f_regression(df[['TV','Radio']],df[['sales']])[0],
... zip(f_regression(df[['TV','Radio']],df[['sales']])[1],lr.predict(df[['TV','Radio']])[0:5,0]*1000))
[(('TV', 0.04575482), (859.58275449, (8.3557004000000000e-112, 9.39426753)),
 (('Newspapers', -0.00103749), (548.20991694, (6.3946766700000000e-086, 7.62085875)),
 (('sales', 0.18799423), (inf, (0.0, 7.90268321)),
 (('Radio', 1.0000000000000000), (inf, (0.0, 8.35811992)),
 (('Intercept', 2.9210999124016331), (0.0, (0.0, 9.43160054)))
>>>
>>> df.corr()
          TV  Newspapers     sales     Radio
TV  1.000000    0.056648  0.782224  0.054809
Newspapers  0.056648    1.000000  0.228299  0.354104
sales       0.782224    0.228299  1.000000  0.576748
Radio       0.054809    0.354104  0.576748  1.000000
>>>
>>> df[['TV','Radio','Newspapers']].corrwith(df[['sales']])
TV            0.782224
Radio         0.576748
Newspapers    0.228299
dtype: float64
>>>
>>> ggplot(df[['TV','Radio','Newspapers','sales']],\
...     aes(x='TV',y='Radio',size='Newspapers',color='sales')) +\
...     geom_point()
```



Looking at the chart you can tell that most high sales are coming from tv and radio.

Step 5: Hypothesis testing and p-values

Use the Ord hypothesis (let's assume there is a relationship between tv and sales. Now let's test the hypothesis, we start open the hypothesis that there is no relationship between tv and sales.

```
>>> f_regression(df[['TV','Radio','Newspapers']],df[['sales']])
(array([ 859.58275449,  548.20991694,   19.81035911]), array([ 8.35570040e-112,  6.39467667e-086,
>>> zip(zip(df.columns,lr.coef_),zip(f_regression(df[['TV','Radio','Newspapers']],df[['sales']])[0],
... zip(f_regression(df[['TV','Radio','Newspapers']],df[['sales']])[1],lr.predict(df[['TV','Radio','Newspapers']])[0:5,0]*1000))
[(('TV', 0.04575482), (859.58275449, (8.3557004000000000e-112, 9.39426753)),
 (('Newspapers', -0.00103749), (548.20991694, (6.3946766700000000e-086, 7.62085875)),
 (('sales', 0.18799423), (inf, (0.0, 7.90268321)),
 (('Radio', 1.0000000000000000), (inf, (0.0, 8.35811992)),
 (('Intercept', 2.9210999124016331), (0.0, (0.0, 9.43160054)))
>>>
>>> df.corr()
          TV  Newspapers     sales     Radio
TV  1.000000    0.056648  0.782224  0.054809
Newspapers  0.056648    1.000000  0.228299  0.354104
sales       0.782224    0.228299  1.000000  0.576748
Radio       0.054809    0.354104  0.576748  1.000000
>>>
>>> df[['TV','Radio','Newspapers']].corrwith(df[['sales']])
TV            0.782224
Radio         0.576748
Newspapers    0.228299
dtype: float64
>>>
>>> ggplot(df[['TV','Radio','Newspapers','sales']],\
...     aes(x='TV',y='Radio',size='Newspapers',color='sales')) +\
...     geom_point()
```


Looking at the chart you can tell that most high sales are coming from tv and radio.

Step 6: Multiple Linear Regression

Create an new DF for the feature volumn, name it df_features and use the dataset sales to build a multiple linear regression model.

```
>>> df_features = df[['TV','Radio','Newspapers']]
>>> lr = LinearRegression()
>>> lr.fit(df_features,df[['sales']])
LinearRegression(copy_X=True, fit_intercept=True, normalize=False)
>>> lr.coef_
array([[ 0.04575482,  0.18799423,  0.00103749]])
>>> lr.intercept_
array([ 2.92109991])
>>> lr.score(df_features,df[['sales']])
0.897194692937763
>>> f_regression(df_features,df[['sales']])
(array([ 859.58275449,  548.20991694,   19.81035911]), array([ 8.35570040e-112,  6.39467667e-086,
>>> zip(zip(df.columns,lr.coef_),zip(f_regression(df_features,df[['sales']])[0],
... zip(f_regression(df_features,df[['sales']])[1],lr.predict(df_features)[0:5,0]*1000))
[(('TV', 0.04575482), (859.58275449, (8.3557004000000000e-112, 9.39426753)),
 (('Newspapers', -0.00103749), (548.20991694, (6.3946766700000000e-086, 7.62085875)),
 (('sales', 0.18799423), (inf, (0.0, 7.90268321)),
 (('Radio', 1.0000000000000000), (inf, (0.0, 8.35811992)),
 (('Intercept', 2.9210999124016331), (0.0, (0.0, 9.43160054)))
>>>
>>> df.corr()
          TV  Newspapers     sales     Radio
TV  1.000000    0.056648  0.782224  0.054809
Newspapers  0.056648    1.000000  0.228299  0.354104
sales       0.782224    0.228299  1.000000  0.576748
Radio       0.054809    0.354104  0.576748  1.000000
>>>
>>> df[['TV','Radio','Newspapers']].corrwith(df[['sales']])
TV            0.782224
Radio         0.576748
Newspapers    0.228299
dtype: float64
>>>
>>> ggplot(df[['TV','Radio','Newspapers','sales']],\
...     aes(x='TV',y='Radio',size='Newspapers',color='sales')) +\
...     geom_point()
```


Looking at the chart you can tell that most high sales are coming from tv and radio.

Step 7: Importing Libraries

```
>>> import numpy as np
>>> import pandas as pd
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
>>> from math import sqrt
>>> from ggplot import *
>>> from sklearn.preprocessing import Imputer
>>> from sklearn.pipeline import Pipeline
>>> from sklearn.cross_validation import cross_val_score, KFold
>>> from sklearn.grid_search import GridSearchCV
>>> from sklearn.ensemble import RandomForestRegressor
>>> from sklearn.tree import DecisionTreeRegressor
>>> from sklearn.svm import SVR
>>> from sklearn.ensemble import ExtraTreesRegressor, GradientBoostingRegressor
>>> from sklearn.feature_selection import f_regression
>>> from sklearn.ensemble.partial_dependence import partial_dependence, plot_partial_dependence
>>> from mpl_toolkits.mplot3d import Axes3D
>>> from StringIO import StringIO
>>> from sklearn.externals.joblib import Memory
>>> import warnings
>>> warnings.filterwarnings('ignore')
>>> %pylab inline
Populating the interactive namespace from numpy and matplotlib
WARNING: pylab import has clobbered these variables: ['ceil', 'sqrt', 'exp', 'e', 'log', 'pi']
>>> %load_ext rmagic
>>> %load_ext ipy_table
>>> %load_ext ipy_table_widgets
>>> %load_ext ipy_table_utils
>>> %load_ext ipy_table_extensions
>>> %load_ext ipy_table_formatters
>>> %load_ext ipy_table_validators
>>> %load_ext ipy_table_converters
>>> %load_ext ipy_table_sparklines
>>> %load_ext ipy_table_cleaners
>>> %load_ext ipy_table_sources
>>> %load_ext ipy_table_filters
>>> %load_ext ipy_table_sorters
>>> %load_ext ipy_table_widgets_ipy
>>> %load_ext ipy_table_widgets_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui_ipy_ui
>>> %load_ext ipy_table_widgets_ipy_ui
```

Regularization

Regularization does not improve the performance of the algorithm or the quality of the model but it does improve the loss defined as per the model parameters. It can reduce the generalization performance, i.e. the performance on new cases.

Types of Regression (contd)

Ridge Regression

Lasso Regression

Elastic Net Regression

Demo: Bigmart

Problem Statement: Bigmart has collected sales data for 1559 products across 10 stores in different cities. Attributes of each product and each store have been defined. The aim is to build a predictive model and find out the sales of each product at each store.

Using this model, Bigmart will try to understand the properties of products and stores which play a big role in increasing sales.

what could affect the target variable (sales) ?

1. The size of the store - Sale is expected to be high at a medium SIZE
2. Higher sales according to the LOC of the city that is, visiting and item average.
3. Type of the store - Type 3 is better
4. Fat or regular - Fat is better
5. Visibility area - Sales are better at low visibility area

Objective

- Build a predictive model and find out the sale of each product at a particular store.
- Using Ridge and Lasso regression techniques, interpret the coefficients of the model
- Find out the RMSE and R-square
- Evaluate model efficiency using RMSE and R-Square values

```
import numpy as np
import pandas as pd
import matplotlib as mpl
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, ElasticNetCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_block_info=True,
        color_scheme='Blues',
        rc={'figure.figure_format': 'retina'})
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
pd.set_option('display.max_columns', None)
train = pd.read_csv('Train_UWu5bXk.csv')
test = pd.read_csv('Test_u94Q5KV.csv')
print('Train data shape: ', train.shape)
print('Test data shape: ', test.shape)
```

```
Train data shape:  (8523, 12)
Test data shape:  (5681, 11)
```

```
train.head(5)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	2987	High	Tier 3	Supermarket Type1	994.7052
5	FDH17	10.395	Regular	0.076069	Frozen Foods	51.4008	OUT018	2009	Medium	Tier 3	Supermarket Type2	556.6088
6	FDU28	13.650	Regular	0.054464	Frozen Foods	57.5904	OUT046	1997	Small	Tier 1	Supermarket Type1	343.5528
7	FDY07	11.800	Low Fat	0.044950	Fruits and Vegetables	159.5420	OUT049	1999	Medium	Tier 1	Supermarket Type1	2403.1500
8	FDA03	18.600	Low Fat	0.047257	Dairy	192.1847	OUT045	2002	NaN	Tier 2	Supermarket Type1	1845.5976
9	FDX32	15.100	Regular	0.035186	Fruits and Vegetables	81.5592	OUT010	1998	NaN	Tier 3	Grocery Store	765.6720
10	FDF53	20.750	reg	0.083183	Frozen Foods	186.8344	OUT017	2007	NaN	Tier 2	Supermarket Type1	2587.9640

Item Identifier - Product ID, Not a relevant variable

Item Weight - Weight-quantity gap Outlier

Item Fat Content - Categorical column, ordered data

Item Visibility - Continuous data

Item Type - Text or Alphabet variable

Item MRP - Continuous data

Outlet Identifier - Text or alphabet variable

Outlet Establishment Year - Identify the column by subtracting current year with the establishment year

Outlet Size - Categorical column, ordered

Outlet Location Type - Categorical, ordered

Outlet Type - Categorical, Text variable

Item_Outlet_Sales - Target variable, continuous variable

Name Item_Outlet_Sales in the target variable

```
train.Item_Outlet_Sales.hist(bins=50,figsize=(8,5),color='Green')
plt.suptitle("Histogram of Item_Outlet_Sales",fontsize=15)
plt.xlabel("Item_Outlet_Sales")
plt.ylabel("Count")
plt.show()
```

```
# Filling the missing values
train.isnull().sum()
```

```
Item_Identifier            0
Item_Weight             1463
Item_Fat_Content           0
Item_Visibility            0
Item_Type                  0
Item_MRP                   0
Outlet_Identifier          0
Outlet_Establishment_Year  0
Outlet_Size             2410
Outlet_Location_Type       0
Outlet_Type                0
Item_Outlet_Sales          0
dtype: int64
```

```
test.isnull().sum()
```

```
Item_Identifier            0
Item_Weight             976
Item_Fat_Content           0
Item_Visibility            0
Item_Type                  0
Item_MRP                   0
Outlet_Identifier          0
Outlet_Establishment_Year  0
Outlet_Size             1606
Outlet_Location_Type       0
Outlet_Type                0
dtype: int64
```

```
train.Item_Weight = train.Item_Weight.fillna(train.Item_Weight.mean)
train.Outlet_Size = train.Outlet_Size.fillna('Medium')
test.Item_Weight = test.Item_Weight.fillna(test.Item_Weight.mean)
test.Outlet_Size = test.Outlet_Size.fillna('Medium')
```

```
train.isnull().sum()
```

```
Item_Identifier            0
Item_Weight                0
Item_Fat_Content           0
Item_Visibility            0
Item_Type                  0
Item_MRP                   0
Outlet_Identifier          0
Outlet_Establishment_Year  0
Outlet_Size                0
Outlet_Location_Type       0
Outlet_Type                0
Item_Outlet_Sales          0
dtype: int64
```

```
train.Item_Fat_Content.value_counts(normalize=True)*100
```

```
Low Fat    64.529
Regular    35.471
LF          0.352
reg         0.117
low fat     0.117
Name: Item_Fat_Content, dtyp
```

```
train.Item_Fat_Content = train.Item_Fat_Content.replace{'LF':'Low Fat','reg':'Regular','low fat':'Low Fat'}
train.Item_Fat_Content.value_counts(normalize=True)*100
```

```
Low Fat    64.646
Regular    35.354
Name: Item_Fat_Content, dtyp
```

```
train.Outlet_Establishment_Year = 2013 - train.Outlet_Establishment_Year
test.Outlet_Establishment_Year = 2013 - test.Outlet_Establishment_Year
```

```
train.Item_Type.value_counts(normalize=True)*100
```

```
Fruits and Vegetables    13.136
Snack Foods              13.047
Household                 9.104
Frozen Foods              8.192
Dairy                     7.207
Canned                    7.160
Baking Goods              7.031
Health and Hygiene        6.178
Soft Drinks               5.449
Meat                      4.372
Breakfast                 2.348
Hard Drinks               1.766
Others                    1.369
Starchy Foods             0.514
Seafood                   0.391
Name: Item_Type, dtype: f
```

```
train.Outlet_Size.value_counts(normalize=True)*100
```

```
Medium    62.676
Small     24.477
High      12.847
Name: Outlet_Size, dtype: f
```

```
train.Outlet_Type.value_counts(normalize=True)*100
```

```
Supermarket Type1    65.371
Grocery Store        21.127
Supermarket Type3     6.503
Supermarket Type2     6.499
Name: Outlet_Type, dtype: f
```

```
train.Outlet_Location_Type.value_counts(normalize=True)*100
```

```
Tier 3    35.238
Tier 2    32.811
Tier 1    31.951
Name: Outlet_Location_Type, dtype: f
```

```
train.Outlet_Size.value_counts(normalize=True)*100
```

```
Medium    62.676
Small     24.477
High      12.847
Name: Outlet_Size, dtype: f
```

```
train.Item_Identifier.value_counts(normalize=True)*100
```

```
FDW13    0.117
FDG33    0.117
NCN55    0.117
FDD38    0.117
DRE49    0.117
FDV60    0.117
FDY43    0.117
FDX02    0.117
FDE52    0.117
FDZ50    0.117
FDP10    0.117
FDN52    0.117
FDK57    0.117
FDA17    0.117
FDS57    0.117
FDR56    0.117
FDR50    0.117
FDP49    0.117
FDN15    0.117
FDM58    0.117
FDB58    0.117
FDX53    0.117
FDC23    0.117
FDB22    0.117
FDU20    0.117
FDT35    0.117
FDC37    0.117
FDB12    0.117
FDA12    0.117
FDF52    0.117
FDW58    0.117
FDU28    0.117
FDY07    0.117
FDA03    0.117
FDX32    0.117
FDF53    0.117
FDH17    0.117
FDN33    0.117
FDY08    0.117
FDD06    0.117
FDW14    0.117
FDG45    0.117
FDU37    0.117
FDB02    0.117
FDN34    0.117
FDG09    0.117
FDB10    0.117
FDV38    0.117
FDD14    0.117
FDG11    0.117
FDN48    0.117
FDD48    0.117
FDB23    0.117
FDU08    0.117
FDX40    0.117
FDV10    0.117
FDU55    0.117
FDX07    0.117
FDX11    0.117
FDA15    0.117
FDX31    0.117
FDU32    0.117
FDU13    0.117
FDT50    0.117
FDN15    0.117
FDX14    0.117
NCF55    0.117
FDW10    0.117
FDV57    0.117
FDA10    0.117
FDW31    0.117
FDU33    0.117
FDV31    0.117
FDU24    0.117
FDT07    0.117
FDC14    0.117
FDG33    0.117
NCL31    0.117
FDE41    0.117
FDD47    0.117
FDV60    0.117
FDB58    0.117
FDD38    0.117
FDP36    0.117
FDK57    0.117
FDW58    0.117
FDJ26    0.117
FDU37    0.117
FDW56    0.117
FDG33    0.117
NCS17    0.117
FDF22    0.117
FDD38    0.117
FDD06    0.117
FDP33    0.117
FDP36    0.117
FDH56    0.117
FDN15    0.117
FDG33    0.117
FDY43    0.117
FDY43    0.117
FDQ60    0.117
FDD38    0.117
FDD06    0.117
FDD06    0.117
FDD06
```

Sigmoid Probability

The probability of the logistic regression is represented by the Sigmoid function (logistic function or the S curve).

$$P(Y) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

β_0 = represents class bias / number of hours studied

β_1 = represents the probability of passing the exam

Accuracy Metrics

	Predicted	
Actual	Not	Yes
Not	TN	FP
Yes	FN	TP

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

= True Positive (TP) + True Negative (TN)

Confusion Matrix

Confusion Matrix for a multi - class classification

Confusion Matrix for a multi - class classification

ROC curve

Computes the model true positive and false positive rates for the model. It plots a curve against

ROC (true-false the ROC Curve)

Measures the extra two dimensional area under the ROC curve.

Demo: Survival of Titanic Passengers

Problem Statement The sinking of the Titanic is one of the biggest maritime disasters in the history, killing 1500 out of 2224 passengers and crew members. In our case study, we will predict which passengers survived and which died. Some groups of people were more likely to survive than others, such as women, children, and the upper class.

Objective

- Use logistic regression to predict the survival of a given passenger based on features, such as sex, age

```
import numpy as np
import pandas as pd
%matplotlib inline
```

```
data_train.shape
```

```
(891, 12)
```

```
data_train.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
30	31	0	1	Uruchurtu, Don. Manuel E	male	40.0	0	0	PC 17601	27.7208	NaN	C
73	74	0	3	Chamberlain, Mr. Frederick	male	28.0	0	0	W./C.7268	9.5875	NaN	S
332	333	0	1	Hamalainen, Mrs. Anna	female	30.0	0	0	250649	13.8583	NaN	S
74	75	1	3	Bing, Mr. Lee	male	32.0	0	0	1601	56.4958	NaN	S
444	445	1	3	Johnson, Master. Harold Theodor	male	4.0	1	1	S.W./PP 752	11.1333	NaN	S

```
data_train
```

```
data_test.shape
```

```
(418, 11)
```

```
data_test.sample(5)
```

```
PassengerId
Survived
Pclass
Name
Sex
Age
SibSp
Parch
Ticket
Fare
Cabin
Embarked
dtype: object
```

```
data = data_train.fillna('Passenger').fillna('Female').fillna('Female').fillna(0).sample(5)
data.head(5)
```

	Survival	Pclass	Sex	Age	Fare	Embarked
0	0	3	Male	22.0	7.2500	S
1	1	1	Female	38.0	71.2833	C
2	1	3	Female	26.0	7.9250	S
3	1	1	Female	35.0	53.1000	S
4	0	3	Male	35.0	8.0500	S

```
data.shape
```

```
(891, 12)
```

```
data.describe()
```

```
Survival
Pclass
Age
SibSp
Parch
Fare
dtype: object
```

```
data.describe(include=['object'])
```

```
Sex
Embarked
dtype: object
```

```
data.info()
```

```
Survival
Pclass
Sex
Age
Fare
Embarked
dtype: object
```

```
data.head()
```

	Survival	Pclass	Sex	Age	Fare	Embarked
0	0	3	Male	22.0	7.2500	S
1	1	1	Female	38.0	71.2833	C
2	1	3	Female	26.0	7.9250	S
3	1	1	Female	35.0	53.1000	S
4	0	3	Male	35.0	8.0500	S