Workshop Week 2

March 7, 2021

1 Supervised Learning: Regression Modeling

In this week's workshop, we will learn about `supervised machine learning` wherein an annotated data having input-output pair is given to us to train machine learning model. The trained model then can be used to make predictions on `unsee` data (or test set). We aim to work on the following tasks: - Ordinary linear regression model on one-dimensional feature/variable - Ordinary linear regression model on multi-dimensional features/variables - Ridge regression and hyper-parameter tuning

1.0.1 Dataset Description

In the `data` directory, we are given `houses_portland.csv` file which contains columns for `area of a house`, `number of bedrooms in a house`, and `price`. The goal is to build model which can predict `price of a house` given `number of bedrooms` and `area`. First we will load dataset and find relationship between variables.

[2]:
```
# Import the commonly-used modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[3]:
```
# Load the dataset
data = pd.read_csv("/home/dalia/Master of Information Technology in␣
 ↪Cybersecurity/Session 3/COMP8325 Artificial Intelligence/week_2/Week 2␣
 ↪Workshop-20210301/data/houses_portland.csv")
print("data size: "+str(data.shape))
data.head()
```

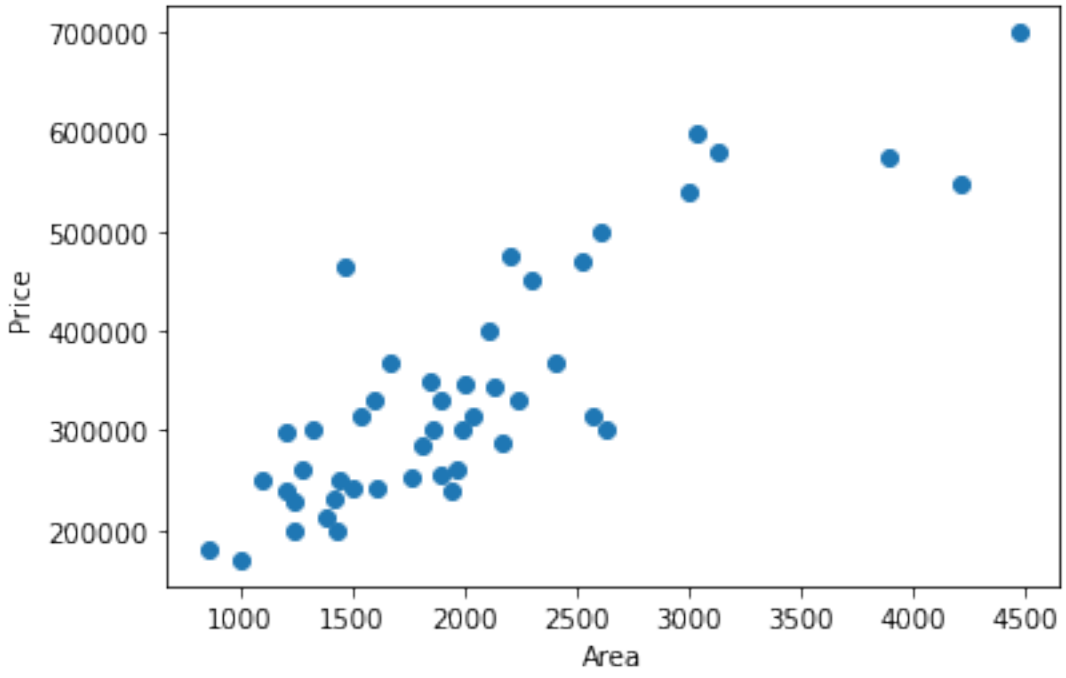
```
data size: (47, 3)
```

[3]:

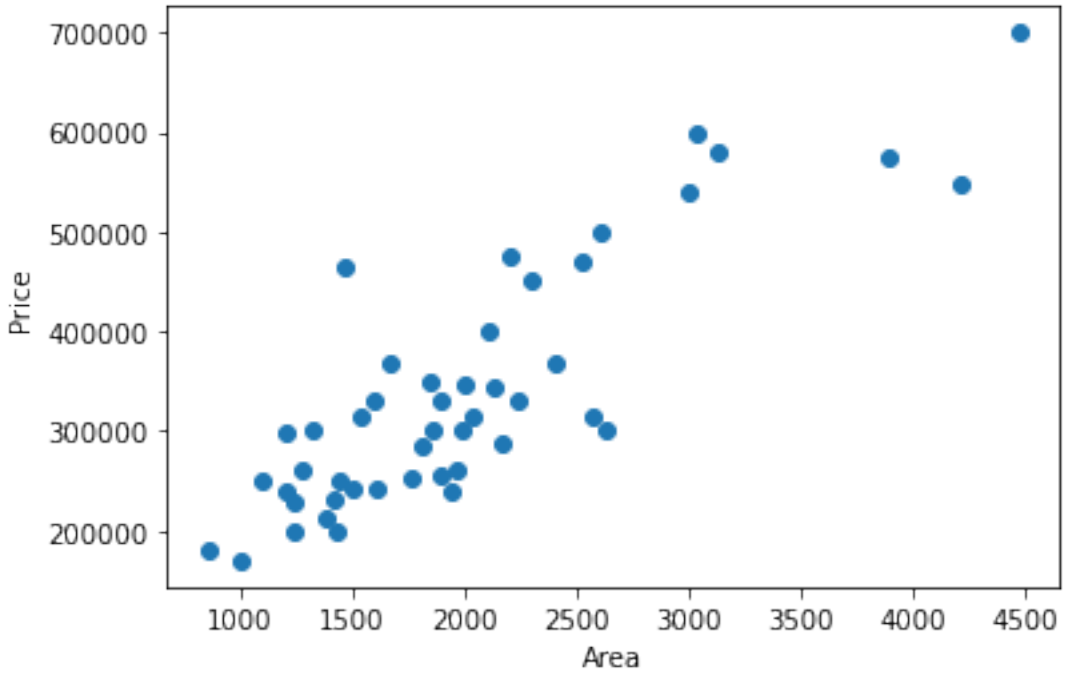
	area	bedroom	price
0	2104	3	399900
1	1600	3	329900
2	2400	3	369000
3	1416	2	232000
4	3000	4	539900

1.1 Task 1: Ordinary linear regression model on one-dimensional feature/variable

[4]:
```
# Visually explore the relationship between "area" and "price"
X = data['area'].values
y = data['price'].values
plt.scatter(X, y)
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```



[5]:
```
# Visually explore the relationship between "area" and "price"
X = data['area'].values
y = data['price'].values
plt.scatter(X, y)
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```



1.1.1 Split data into train and test using train_test_split method from sklearn library.

[6]:
```
# Split data
from sklearn import model_selection as ms
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=0.2)
X_train = np.reshape(X_train, (-1, 1))  # change from 1D array to 2D array
X_test = np.reshape(X_test, (-1, 1))
print("training data feature shape:", X_train.shape)
print("training data label shape:", y_train.shape)
print("test data feature space shape:", X_test.shape)
print("test data label shape:", y_test.shape)
```

```
training data feature shape: (37, 1)
training data label shape: (37,)
test data feature space shape: (10, 1)
test data label shape: (10,)
```

[7]:
```
# Split data
from sklearn import model_selection as ms
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=0.2)
X_train = np.reshape(X_train, (-1, 1))  # change from 1D array to 2D array
X_test = np.reshape(X_test, (-1, 1))
print("training data feature shape:", X_train.shape)
```

```
print("training data label shape:", y_train.shape)
print("test data feature space shape:", X_test.shape)
print("test data label shape:", y_test.shape)
```

```
training data feature shape: (37, 1)
training data label shape: (37,)
test data feature space shape: (10, 1)
test data label shape: (10,)
```

Looking at the shape of train and test sets, we can say that out of total 47 samples, 37 samples are in training and the remaining 10 samples are in test. Look at the documentation of `train_test_split` function to find out whether data is shuffled or not during the split operation.

1.1.2 Model Development (or model training)

[8]:
```
# Build linear regression model
from sklearn import linear_model as lm
ordinaryLRmodel = lm.LinearRegression()
ordinaryLRmodel.fit(X_train, y_train)
```

[8]:
```
LinearRegression()
```

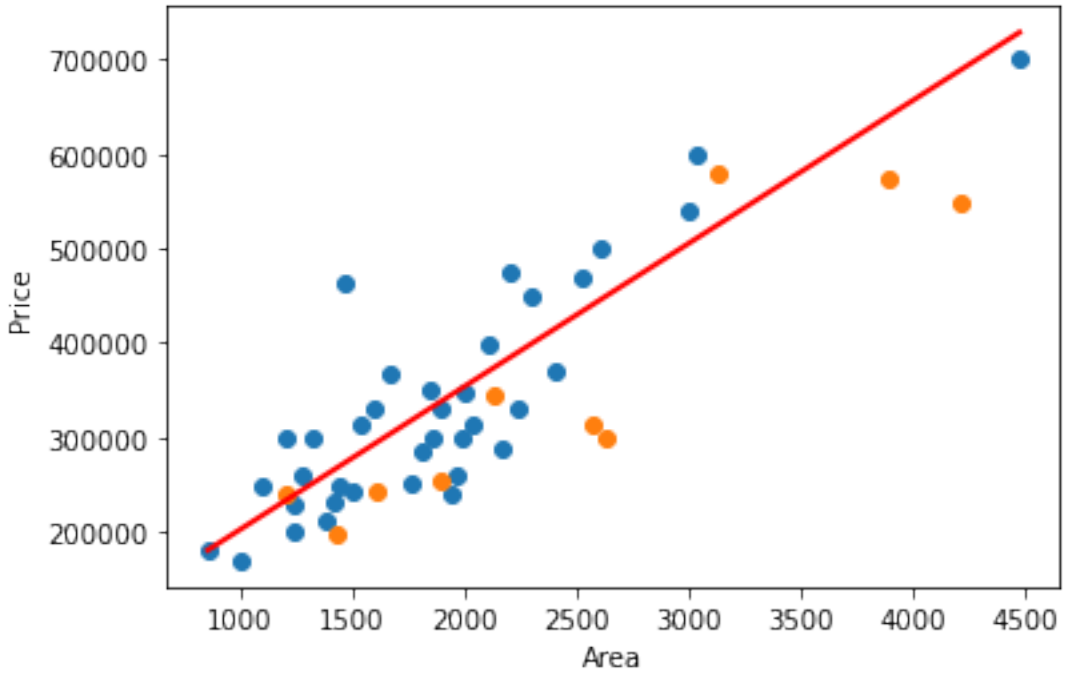
[9]:
```
# Print the coefficients
print("coefficient:", ordinaryLRmodel.coef_)
print("intercept:", ordinaryLRmodel.intercept_)
```

```
coefficient: [150.98324043]
intercept: 52252.05335751246
```

What is an Intercept? How you define coefficient? Can you comment on the importance of coefficient and intercept? What does these numbers signify in terms of price of a house?

1.1.3 Plotting regression line

[10]:
```
# Plot the model
x_line=[X.min(), X.max()]
y_line=[ordinaryLRmodel.intercept_+ordinaryLRmodel.coef_*X.min(),␣
 ↪ordinaryLRmodel.intercept_+ordinaryLRmodel.coef_*X.max()]
plt.plot(x_line, y_line, 'r', lw=2)
plt.scatter(X_train, y_train)
plt.scatter(X_test, y_test)
plt.xlabel('Area')
plt.ylabel('Price')
plt.show()
```



1.1.4 Model Evaluation (on test set)

[11]:
```
# Prediction/Testing
y_pred = ordinaryLRmodel.predict(X_test)
y_pred
```

[11]:
```
array([439826.03155166, 294429.17101369, 374148.32196286, 233884.89159965,
       688646.41178694, 525886.47859906, 450394.85838204, 337308.41129696,
       267705.13745687, 639576.85864588])
```

[12]:
```
# Performance metrics
from sklearn.metrics import mean_squared_error, r2_score

# The mean squared error
print('Mean squared error: %.2f' %mean_squared_error(y_test, y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f' %r2_score(y_test, y_pred))
```

```
Mean squared error: 8024454067.16
Coefficient of determination: 0.60
```

Comment on the model performance in terms of evaluation metrics?

1.2 Task 2: Ordinary linear regression model on multi-dimensional features/variables

[13]:
```
# Data preparatoin
X = data[['area', 'bedroom']].values
y = data['price'].values

# Split data
print(X.shape)
print(y.shape)
```

```
(47, 2)
(47,)
```

1.2.1 Split data into training and testing sets

[14]:
```
# Write your code here
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=0.2,␣
↪random_state=142)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(37, 2)
(37,)
(10, 2)
(10,)
```

1.2.2 Build linear regression model

[15]:
```
# Write your code here
model = lm.LinearRegression()
model.fit(X_train, y_train)
```

[15]:
```
LinearRegression()
```

1.2.3 Check model coefficients and comment

[16]:
```
# Write your code here
print("Intercept: ", model.intercept_)
print("Coefficient: ", model.coef_)
```

```
Intercept:  63175.63367314107
Coefficient:  [141.80327823 123.16809456]
```

1.2.4 Do model predictions on test set

[17]:
```
# Write your code here
y_pred = model.predict(X_test)
y_pred
```

[17]:
```
array([265898.41598452, 338766.66580873, 243351.69474667, 370246.99357478,
       264215.41182948, 239239.39967814, 238813.98984346, 661369.12377161,
       281071.36675464, 299382.62482939])
```

1.2.5 Get model scores in terms of MSE and r^2 scores

[18]:
```
# Write your code here
print("MSE: ", mean_squared_error(y_test, y_pred))
print("R2 score: ", r2_score(y_test, y_pred))
```

```
MSE:  4262225572.0520506
R2 score:  0.5763783233257809
```

1.3 Task 3: Ridge regression and hyperparameter tuning

[31]:
```
# Data preparatoin
X = data[['area', 'bedroom']].values
y = data['price'].values

# Data standardization
from sklearn.preprocessing import StandardScaler
ss = StandardScaler(with_mean=True, with_std=True)
ss.fit(X.astype(np.float))
X = ss.transform(X.astype(np.float))

y = np.reshape(y, (-1, 1))
ss.fit(y.astype(np.float))
y = ss.transform(y.astype(np.float))
```

1.3.1 Split data into training and test sets

[32]:
```
# Write your code here
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=0.2)
```

1.3.2 Build linear regression model

[34]:
```
# Write your code here
ridgeLRmodel = lm.Ridge(alpha=0)
ridgeLRmodel.fit(X_train, y_train)
```

[34]:
```
Ridge(alpha=0)
```

1.3.3 Check model coefficients

[35]:
```
# Write your code here
print("Coefficient: ", ridgeLRmodel.coef_)
```

```
Coefficient:  [[ 0.86824198 -0.09417262]]
```

1.3.4 Do model predictions on test set

[36]:
```
# Write your code here

y_pred = ridgeLRmodel.predict(X_test)
y_pred
```

[36]:
```
array([[ 0.52047956],
       [-0.08831459],
       [-0.60595629],
       [-0.44111038],
       [-0.76166948],
       [-0.77543016],
       [-1.06396314],
       [-0.36299954],
       [-0.80142604],
       [-0.48006052]])
```

1.3.5 Get model scores in terms of MSE and r^2 scores

[37]:
```
# Write your code here
print('MSE: %.2f' %mean_squared_error(y_test, y_pred))

print('R2 score: %.2f' %r2_score(y_test, y_pred))
```

```
MSE: 0.14
R2 score: 0.38
```

1.3.6 Model hyper-parameter tuning

When you check the documentation of `ridge regression` on sklearn API Ridge Regression, you will find `alpha` as one of the parameter that is a `regularisation` term. Regularization improves the conditioning of the problem and reduces the variance of the estimates. Larger values specify stronger regularization.

Your task is to change `alpha` value and see its impact on model performance. Do you see any noticeable change in model performance with changing `alpha` value. Write your analysis in the cell provided.

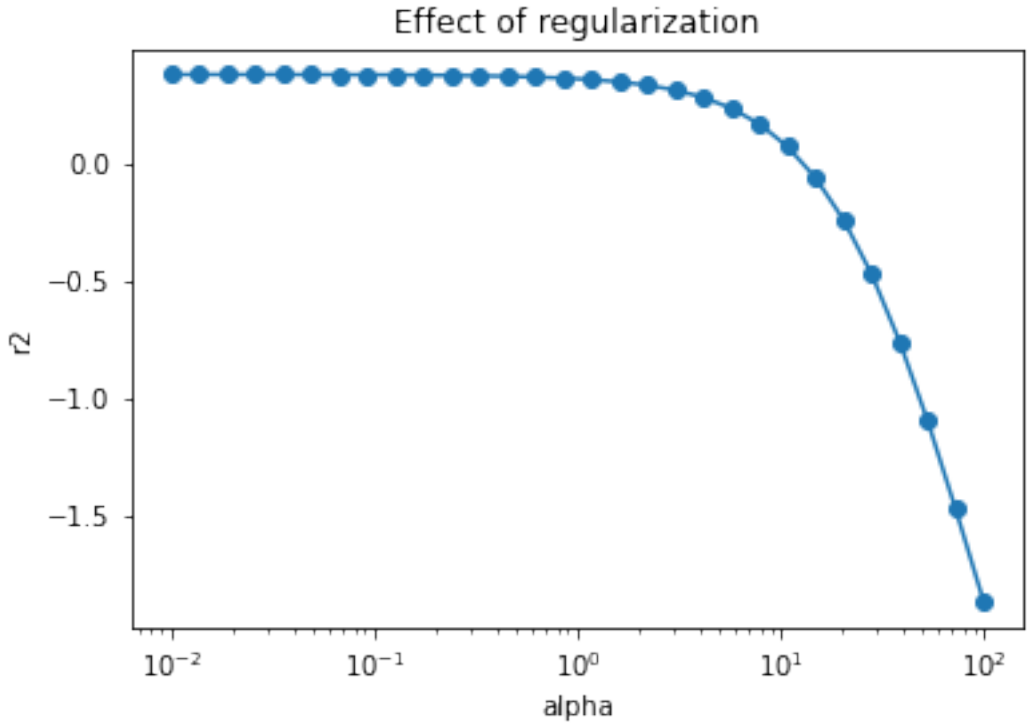
[38]:
```
# Tune the model hyperparameter alpha (See lecture notes)
# Write your code here
```

```
r2s = list()

alphas = np.logspace(-2, 2, 30)

for alpha in alphas:
    ridgeLRmodel = lm.Ridge(alpha=alpha)
    ridgeLRmodel.fit(X_train, y_train)
    y_pred = ridgeLRmodel.predict(X_test)
    r2s.append(r2_score(y_test, y_pred))

plt.plot(alphas, r2s, marker='o')
plt.xscale('log')
plt.title('Effect of regularization')
plt.xlabel('alpha')
plt.ylabel('r2')
plt.show()
```



1.3.7 Write your summary below

T = [[-1, 1], [6, 2], [3, 3],[3, 4], [1, 3], [9, '?']]

[39]:
```
T = [[-1, 1], [6, 2], [3, 3],[3, 4], [1, 3], [9, '?']]
T
```

[39]:
```
[[-1, 1], [6, 2], [3, 3], [3, 4], [1, 3], [9, '?']]
```

[40]:
```
len(T)
```

[40]:
```
6
```

[41]:
```
T = [[-1, 6, 3, 3, 1, 9],[1, 2, 3, 4, 4, '?']]
```

[42]:
```
T
```

[42]:
```
[[-1, 6, 3, 3, 1, 9], [1, 2, 3, 4, 4, '?']]
```

[43]:
```
T[0]
```

[43]:
```
[-1, 6, 3, 3, 1, 9]
```

[44]:
```
T[1]
```

[44]:
```
[1, 2, 3, 4, 4, '?']
```

[45]:
```
def averageOfList(num):
    sumOfNumbers = 0
    for t in num:
        sumOfNumbers = sumOfNumbers + t

    avg = sumOfNumbers / len(num)
    return avg


print("The average of List is", averageOfList(T[0]))
```

```
The average of List is 3.5
```

[46]:
```
T[1]
```

[46]:
```
[1, 2, 3, 4, 4, '?']
```

[47]:
```
print("The average of List is", averageOfList(T[1][:-1]))
```

```
The average of List is 2.8
```

[48]:
```
import statistics as st
```

[49]:
```
st.median(T[0])
```

[49]:
```
3.0
```

[50]:
```
st.mode(T[0])
```

[50]: 3

[51]:
```
st.mean(T[0])
```

[51]: 3.5

[52]:
```
st.mean(T[1][:-1])
```

[52]: 2.8

[53]:
```
from sklearn.preprocessing import normalize

X = [[1, -1, 2], [2, 0, 0], [0, 1, -1]]
```

[54]:
```
l2_norm = normalize(X, norm='l2')
l2_norm
```

[54]:
```
array([[ 0.40824829, -0.40824829,  0.81649658],
       [ 1.        ,  0.        ,  0.        ],
       [ 0.        ,  0.70710678, -0.70710678]])
```

[55]:
```
l1_norm = normalize(X, norm='l1')
l1_norm
```

[55]:
```
array([[ 0.25, -0.25,  0.5 ],
       [ 1.  ,  0.  ,  0.  ],
       [ 0.  ,  0.5 , -0.5 ]])
```

[56]:
```
l2_norm[0]
```

[56]:
```
array([ 0.40824829, -0.40824829,  0.81649658])
```

[57]:
```
l2_norm[0][0]
```

[57]: 0.4082482904638631

[58]:
```
l2_norm[0][1]
```

[58]: -0.4082482904638631

[59]:
```
l2_norm[0][2]
```

[59]: 0.8164965809277261

[60]:
```
import math
```

```
math.sqrt(l2_norm[0][0]*l2_norm[0][0] + l2_norm[0][1] * l2_norm[0][1] +␣
↪l2_norm[0][2] * l2_norm[0][2])
```

[60]: 1.0

[ ]:

[ ]: