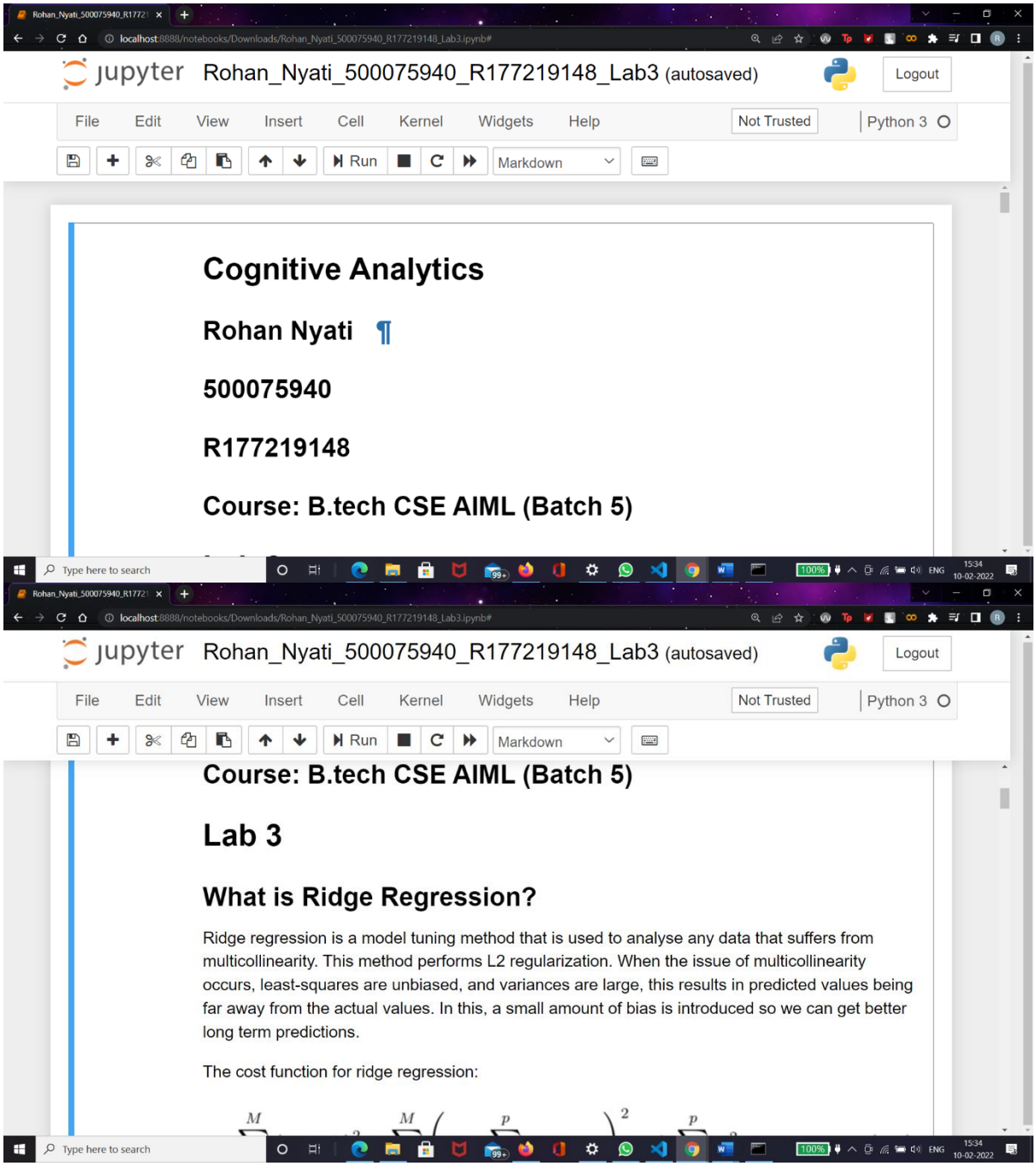
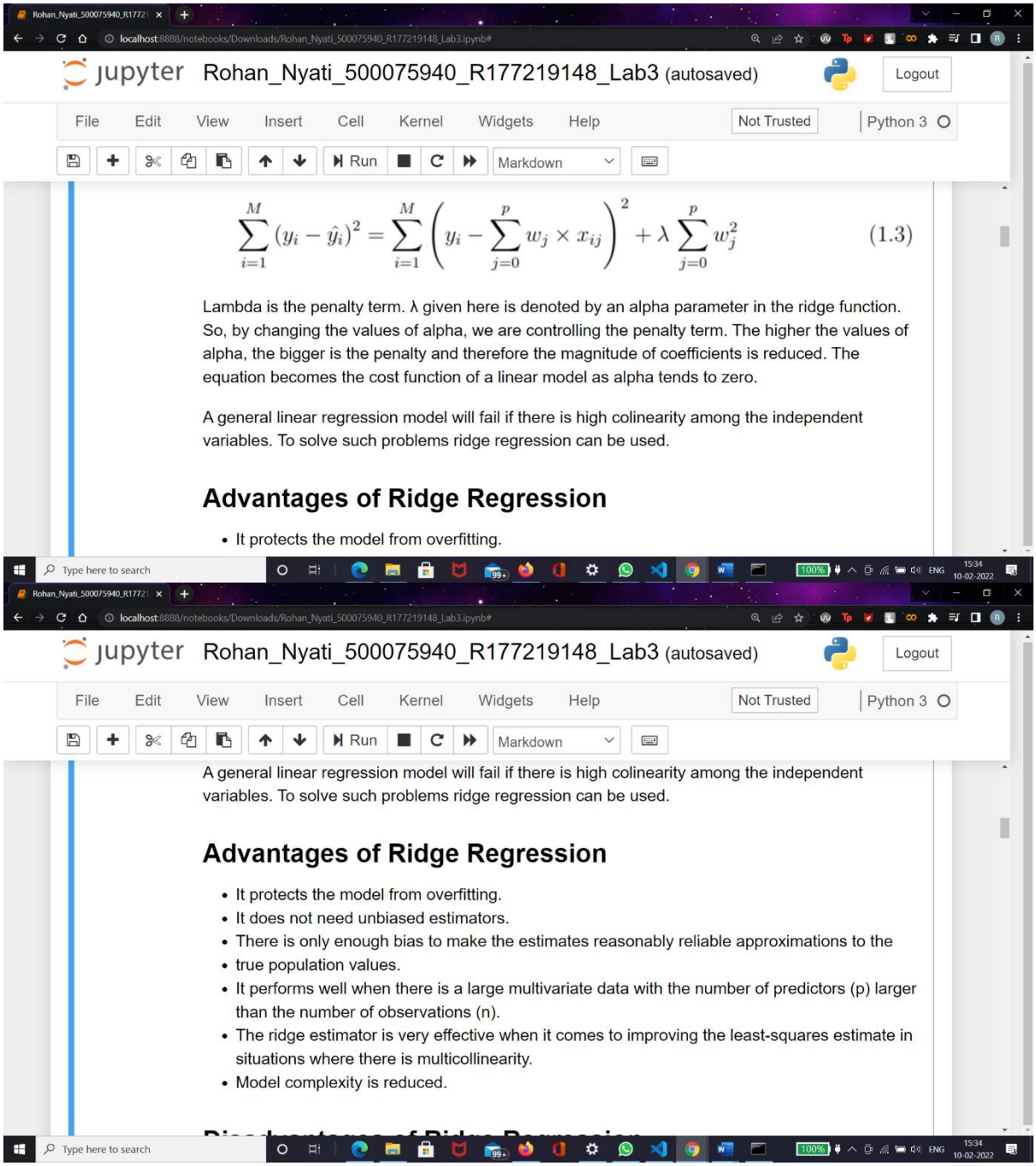
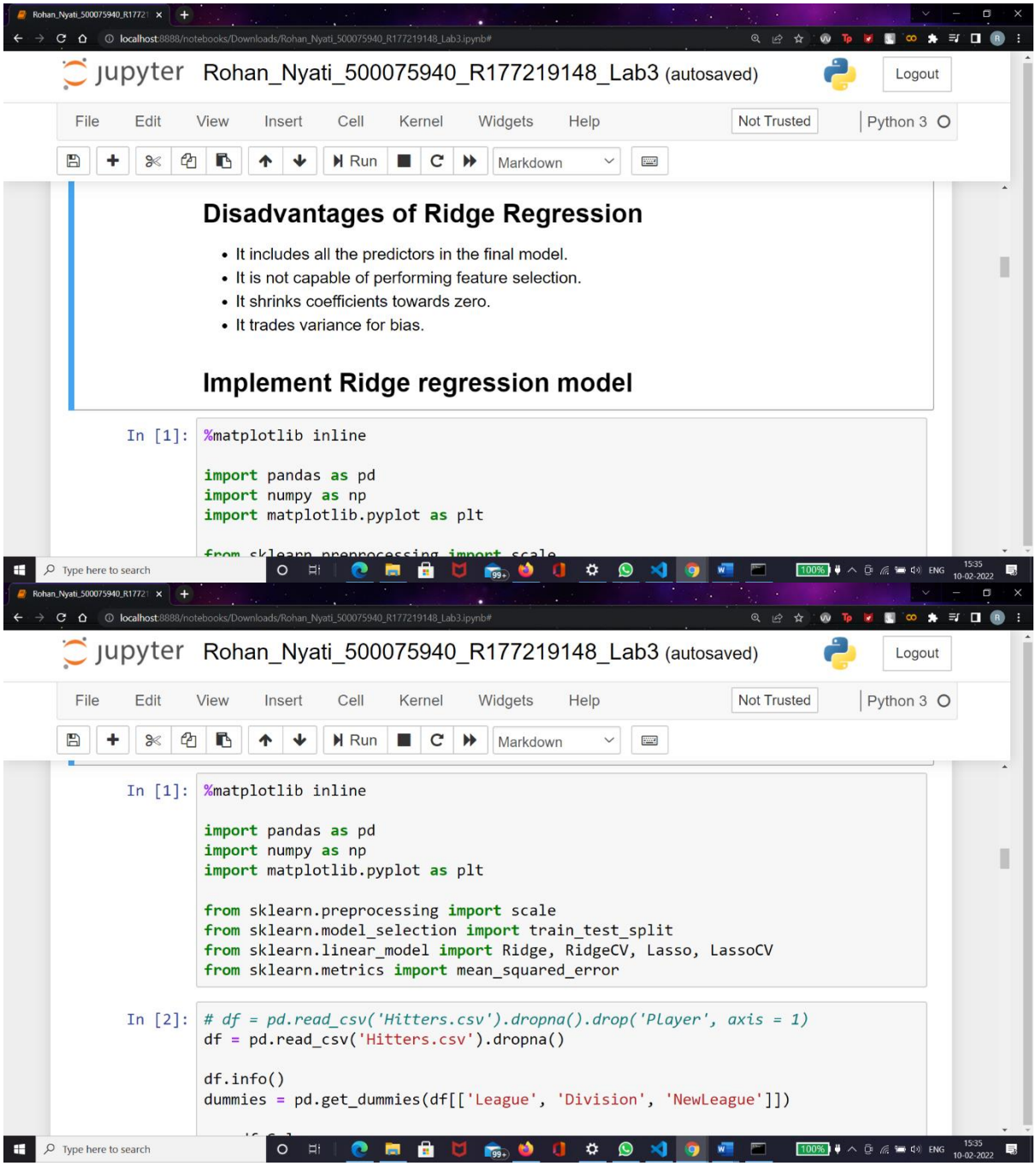
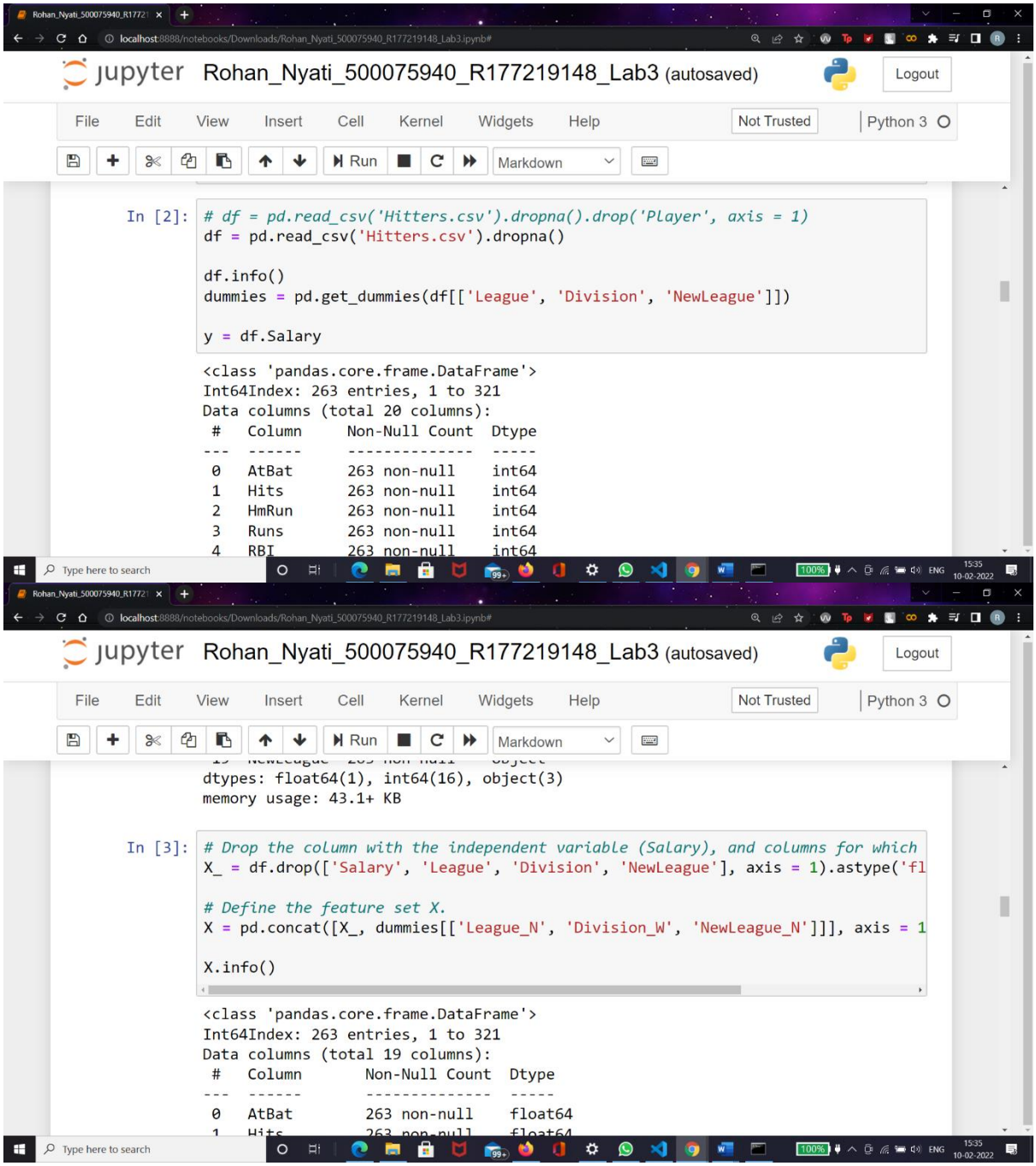
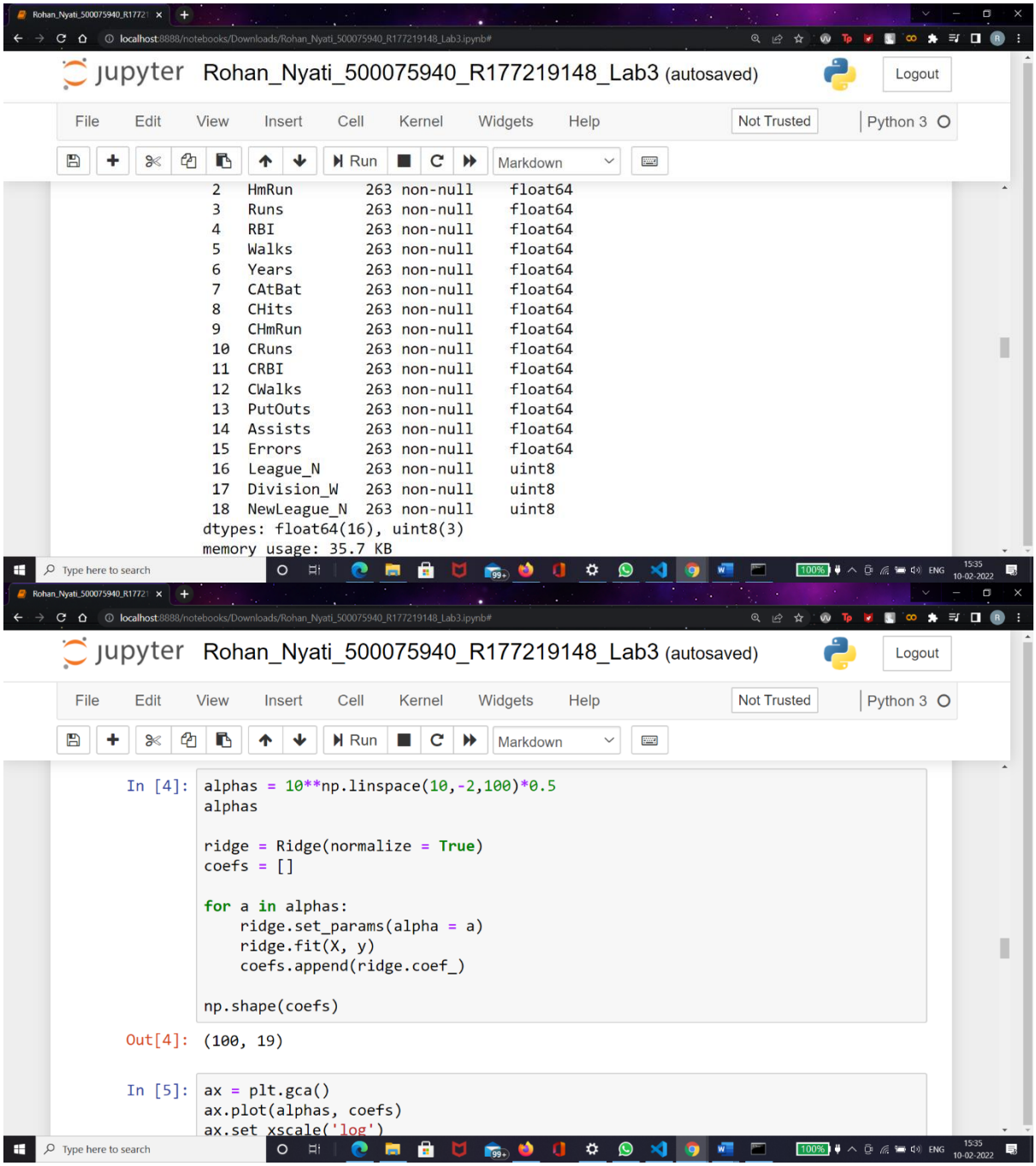

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Cognitive Analytics
Rohan Nyati ¶
500075940
R177219148
Course: B.tech CSE AIML (Batch 5)
Course: B.tech CSE AIML (Batch 5)
Lab 3
What is Ridge Regression?
Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values. In this, a small amount of bias is introduced so we can get better long term predictions.
The cost function for ridge regression:

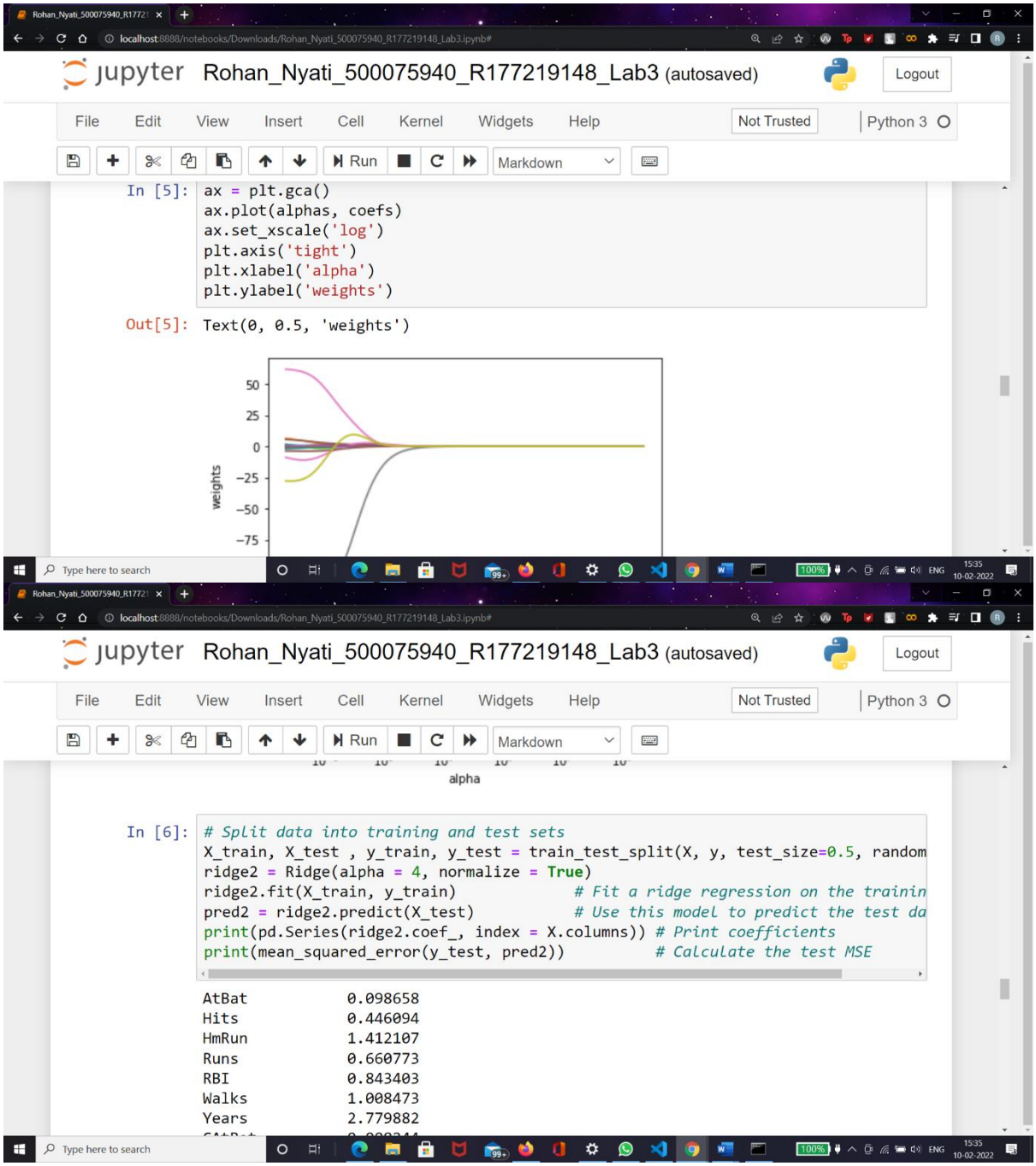

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} w_j^2 \quad (1.3)$$
Lambda is the penalty term. λ given here is denoted by an alpha parameter in the ridge function. So, by changing the values of alpha, we are controlling the penalty term. The higher the values of alpha, the bigger is the penalty and therefore the magnitude of coefficients is reduced. The equation becomes the cost function of a linear model as alpha tends to zero.
A general linear regression model will fail if there is high colinearity among the independent variables. To solve such problems ridge regression can be used.
Advantages of Ridge Regression
It protects the model from overfitting.
Type here to search
ENG
15:34
10-02-2022

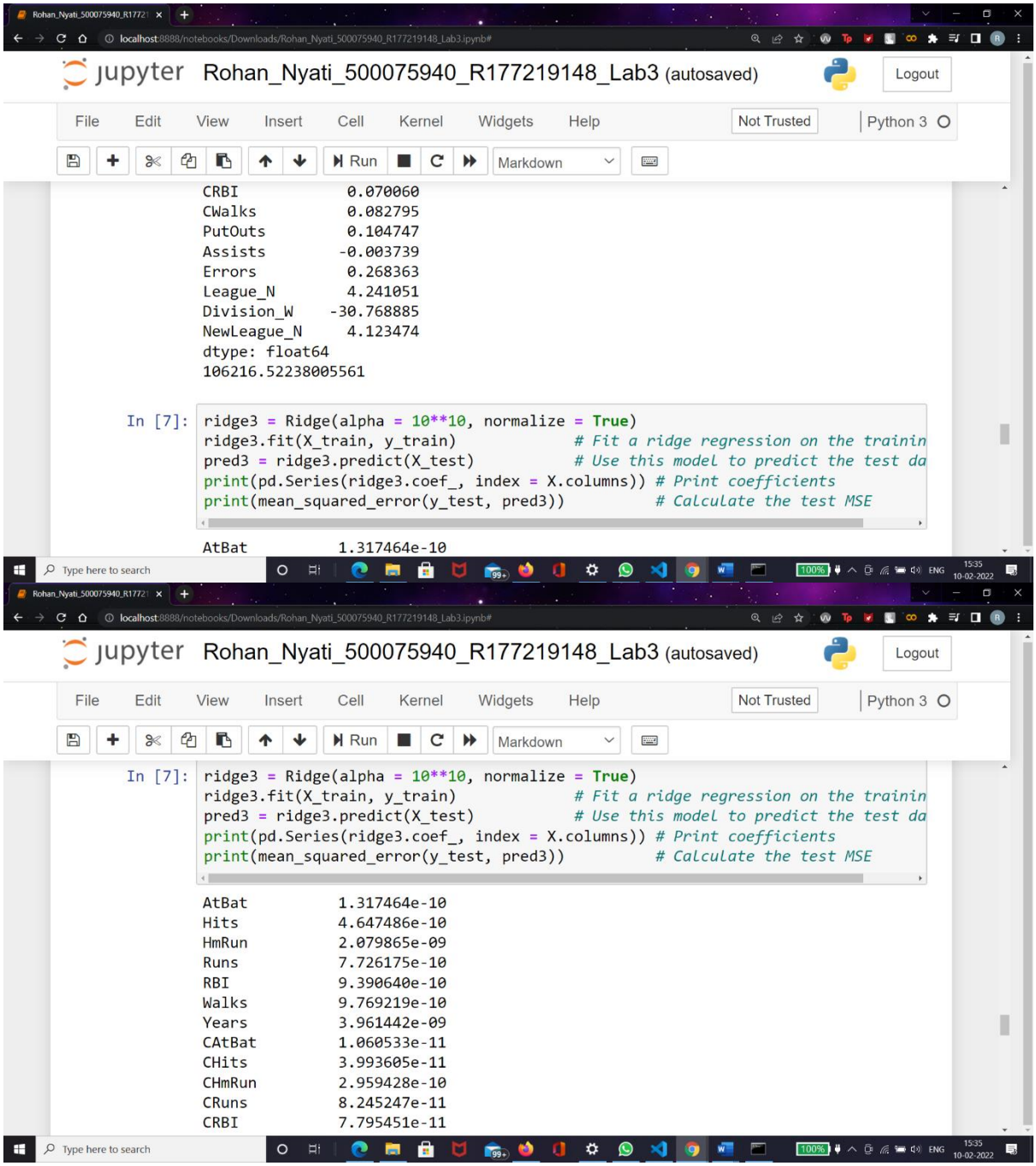
Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
A general linear regression model will fail if there is high colinearity among the independent variables. To solve such problems ridge regression can be used.
Advantages of Ridge Regression
It protects the model from overfitting.
It does not need unbiased estimators.
There is only enough bias to make the estimates reasonably reliable approximations to the
true population values.
It performs well when there is a large multivariate data with the number of predictors (p) larger than the number of observations (n).
The ridge estimator is very effective when it comes to improving the least-squares estimate in situations where there is multicollinearity.
Model complexity is reduced.
Type here to search
ENG
15:34
10-02-2022


Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Disadvantages of Ridge Regression
It includes all the predictors in the final model.
It is not capable of performing feature selection.
It shrinks coefficients towards zero.
It trades variance for bias.
Implement Ridge regression model
In [1]:
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [1]:
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
In [2]:
# df = pd.read_csv('Hitters.csv').dropna().drop('Player', axis = 1)
df = pd.read_csv('Hitters.csv').dropna()
df.info()
dummies = pd.get_dummies(df[['League', 'Division', 'NewLeague']])

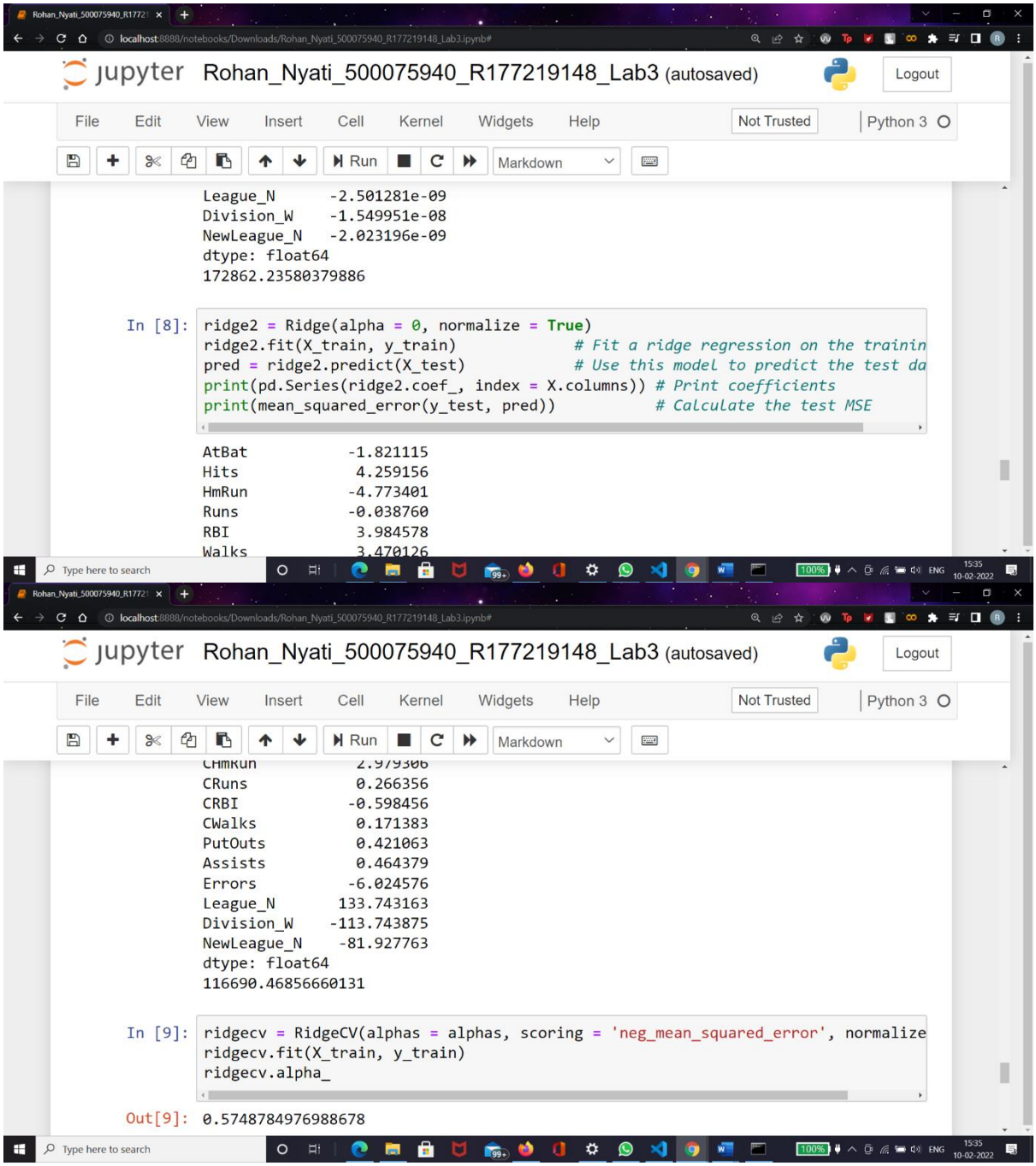

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [2]:
# df = pd.read_csv('Hitters.csv').dropna().drop('Player', axis = 1)
df = pd.read_csv('Hitters.csv').dropna()
df.info()
dummies = pd.get_dummies(df[['League', 'Division', 'NewLeague']])
y = df.Salary
<class 'pandas.core.frame.DataFrame'>
Int64Index: 263 entries, 1 to 321
Data columns (total 20 columns):
Column Non-Null Count Dtype
0 AtBat 263 non-null int64
1 Hits 263 non-null int64
2 HmRun 263 non-null int64
3 Runs 263 non-null int64
4 RBI 263 non-null int64
Type here to search
Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
dtypes: float64(1), int64(16), object(3)
memory usage: 43.1+ KB
In [3]:
# Drop the column with the independent variable (Salary), and columns for which
X_ = df.drop(['Salary', 'League', 'Division', 'NewLeague'], axis = 1).astype('fl
# Define the feature set X.
X = pd.concat([X_, dummies[['League_N', 'Division_W', 'NewLeague_N']]], axis = 1
X.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 263 entries, 1 to 321
Data columns (total 19 columns):
Column Non-Null Count Dtype
0 AtBat 263 non-null float64
Type here to search


Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
2 HmRun 263 non-null float64
3 Runs 263 non-null float64
4 RBI 263 non-null float64
5 Walks 263 non-null float64
6 Years 263 non-null float64
7 CAtBat 263 non-null float64
8 CHits 263 non-null float64
9 CHmRun 263 non-null float64
10 CRuns 263 non-null float64
11 CRBI 263 non-null float64
12 CWalks 263 non-null float64
13 PutOuts 263 non-null float64
14 Assists 263 non-null float64
15 Errors 263 non-null float64
16 League_N 263 non-null uint8
17 Division_W 263 non-null uint8
18 NewLeague_N 263 non-null uint8
dtypes: float64(16), uint8(3)
memory usage: 35.7 KB
Type here to search
Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [4]:
alphas = 10**np.linspace(10,-2,100)*0.5
alphas
ridge = Ridge(normalize = True)
coefs = []
for a in alphas:
ridge.set_params(alpha = a)
ridge.fit(X, y)
coefs.append(ridge.coef_)
np.shape(coefs)
Out[4]: (100, 19)
In [5]:
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
Type here to search

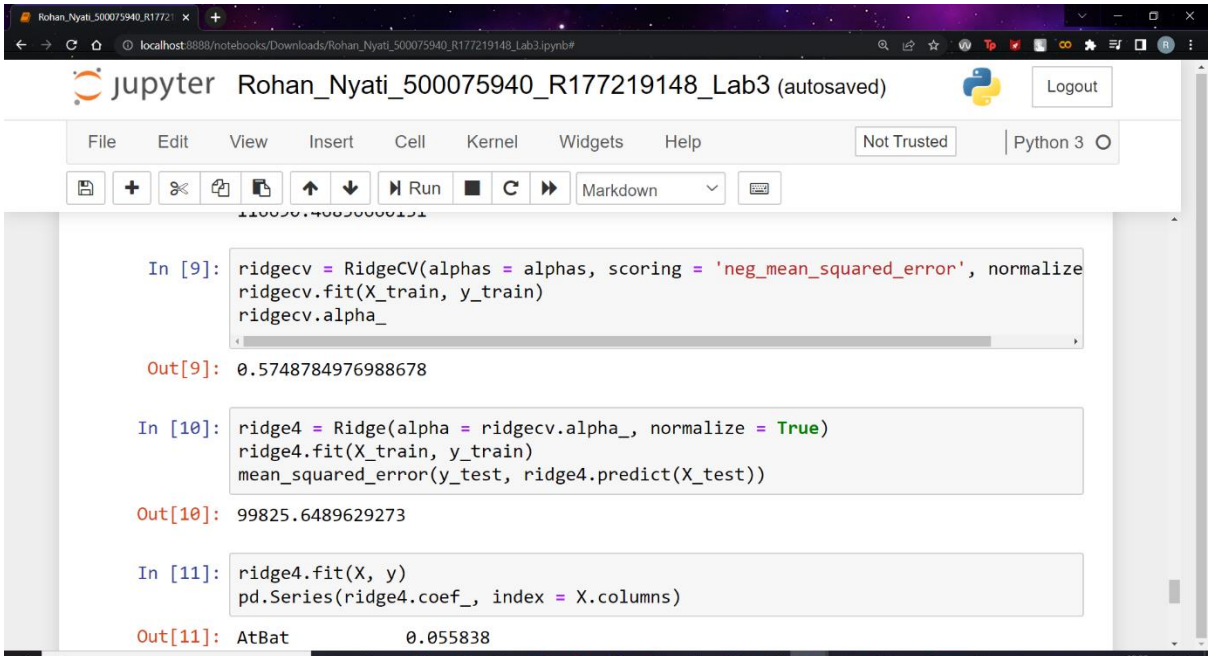

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [5]:
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
Out[5]:
Text(0, 0.5, 'weights')
50
25
0
−25
−50
−75
weights
alpha
In [6]:
Split data into training and test sets
X_train, X_test , y_train, y_test = train_test_split(X, y, test_size=0.5, random
ridge2 = Ridge(alpha = 4, normalize = True)
ridge2.fit(X_train, y_train)
# Fit a ridge regression on the trainin
pred2 = ridge2.predict(X_test)
# Use this model to predict the test da
print(pd.Series(ridge2.coef_, index = X.columns)) # Print coefficients
print(mean_squared_error(y_test, pred2))
# Calculate the test MSE
AtBat 0.098658
Hits 0.446094
HmRun 1.412107
Runs 0.660773
RBI 0.843403
Walks 1.008473
Years 2.779882
Type here to search
ENG
15:35
10-02-2022

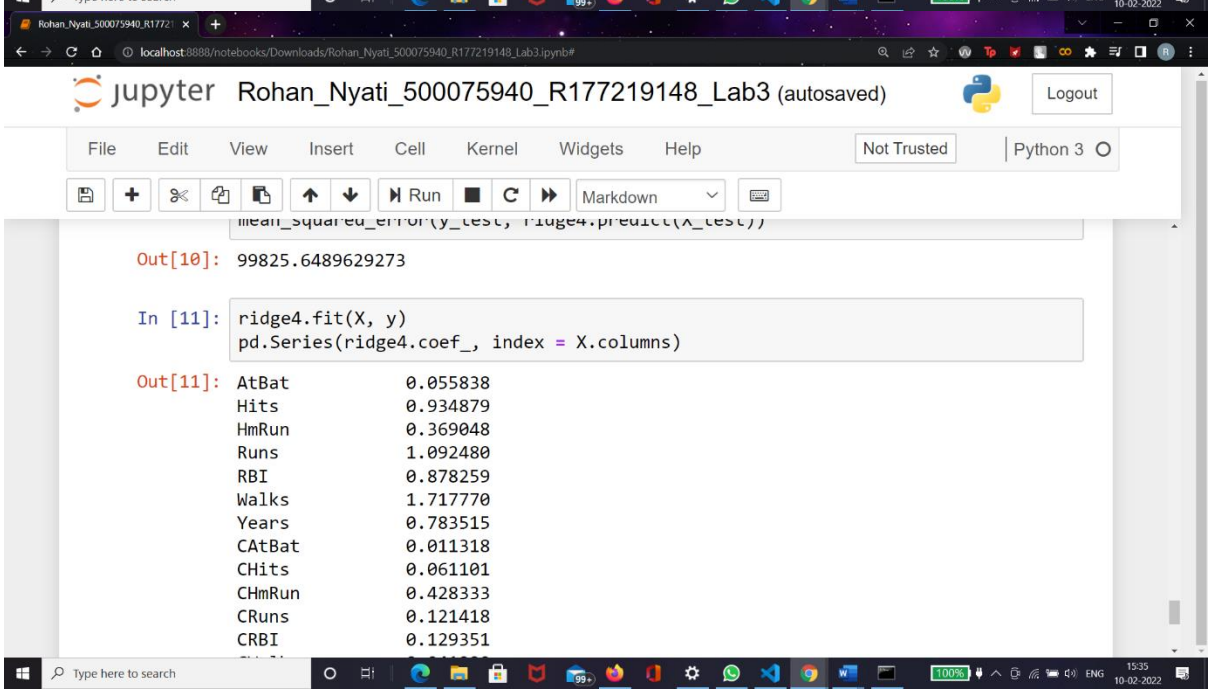

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
CRBI 0.070060
CWalks 0.082795
PutOuts 0.104747
Assists -0.003739
Errors 0.268363
League_N 4.241051
Division_W -30.768885
NewLeague_N 4.123474
dtype: float64
106216.52238005561
In [7]:
ridge3 = Ridge(alpha = 10**10, normalize = True)
ridge3.fit(X_train, y_train) # Fit a ridge regression on the trainin
pred3 = ridge3.predict(X_test) # Use this model to predict the test da
print(pd.Series(ridge3.coef_, index = X.columns)) # Print coefficients
print(mean_squared_error(y_test, pred3)) # Calculate the test MSE
AtBat 1.317464e-10
Type here to search
ENG
15:35
10-02-2022

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [7]:
ridge3 = Ridge(alpha = 10**10, normalize = True)
ridge3.fit(X_train, y_train) # Fit a ridge regression on the trainin
pred3 = ridge3.predict(X_test) # Use this model to predict the test da
print(pd.Series(ridge3.coef_, index = X.columns)) # Print coefficients
print(mean_squared_error(y_test, pred3)) # Calculate the test MSE
AtBat 1.317464e-10
Hits 4.647486e-10
HmRun 2.079865e-09
Runs 7.726175e-10
RBI 9.390640e-10
Walks 9.769219e-10
Years 3.961442e-09
CAtBat 1.060533e-11
CHits 3.993605e-11
CHmRun 2.959428e-10
CRuns 8.245247e-11
CRBI 7.795451e-11
Type here to search
ENG
15:35
10-02-2022


Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
League_N -2.501281e-09
Division_W -1.549951e-08
NewLeague_N -2.023196e-09
dtype: float64
172862.23580379886
In [8]:
ridge2 = Ridge(alpha = 0, normalize = True)
ridge2.fit(X_train, y_train) # Fit a ridge regression on the trainin
pred = ridge2.predict(X_test) # Use this model to predict the test da
print(pd.Series(ridge2.coef_, index = X.columns)) # Print coefficients
print(mean_squared_error(y_test, pred)) # Calculate the test MSE
AtBat -1.821115
Hits 4.259156
HmRun -4.773401
Runs -0.038760
RBI 3.984578
Walks 3.470126
Type here to search
ENG
15:35
10-02-2022

Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
CHmRun 2.979306
CRuns 0.266356
CRBI -0.598456
CWalks 0.171383
PutOuts 0.421063
Assists 0.464379
Errors -6.024576
League_N 133.743163
Division_W -113.743875
NewLeague_N -81.927763
dtype: float64
116690.46856660131
In [9]:
ridgecv = RidgeCV(alphas = alphas, scoring = 'neg_mean_squared_error', normalize
ridgecv.fit(X_train, y_train)
ridgecv.alpha_
Out[9]: 0.5748784976988678
Type here to search
ENG
15:35
10-02-2022


Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [9]:
ridgecv = RidgeCV(alphas = alphas, scoring = 'neg_mean_squared_error', normalize
ridgecv.fit(X_train, y_train)
ridgecv.alpha_
Out[9]: 0.5748784976988678
In [10]:
ridge4 = Ridge(alpha = ridgecv.alpha_, normalize = True)
ridge4.fit(X_train, y_train)
mean_squared_error(y_test, ridge4.predict(X_test))
Out[10]: 99825.6489629273
In [11]:
ridge4.fit(X, y)
pd.Series(ridge4.coef_, index = X.columns)
Out[11]: AtBat 0.055838
Type here to search


Rohan_Nyati_500075940_R177219148_Lab3 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Out[10]: 99825.6489629273
In [11]:
ridge4.fit(X, y)
pd.Series(ridge4.coef_, index = X.columns)
Out[11]:
AtBat 0.055838
Hits 0.934879
HmRun 0.369048
Runs 1.092480
RBI 0.878259
Walks 1.717770
Years 0.783515
CAtBat 0.011318
CHits 0.061101
CHmRun 0.428333
CRuns 0.121418
CRBI 0.129351
Type here to search