NAME OF THE PROJECT

RATINGS PREDICTION

Submitted by:

Bishwajit Bhattacharya

ACKNOWLEDGMENT

I taken the data from Flipkart, although the rating was in decimal value but to get proper output, I round up the data. As a result, the target value is integer.

Most of rating is 4 as because I got the data is mostly like 4.2,4.3,4.4 like this so after round up its coming 4.

From sk learn I got all the regressor model.

INTRODUCTION

Customer reviews are very important part of production. But the problem is how we will know product is likeable or not. By rating a product is median to be sure the product is good or bad.
Normally this rating is done out of 5 some times out of 10 also.
Product quality can be easily measured by this.

Analytical Problem Framing

Data taken from Flipkart website. I used both selenium and Beautiful soup. First data taken from 1st page, there I get 40 data, then tried second page and then all the pages.

For getting proper link I use define function
After I got all the data, I use beautiful soup program.
I taken only model name, star, review and number of ratings.

```
results = soup.find_all('a',{'class':"_1fQZEK"})
model = item.find('div',{'class':"_4rR01T"}).text
star = item.find('div',{'class':"_3LWZlK"}).text
num_ratings = item.find('span',{'class':"_2_R_DZ"}).text.replace('\xa0&\xa0'," ;
")[0:item.find('span',{'class':"_2_R_DZ"}).text
reviews = item.find('span',{'class':"_2_R_DZ"}).text.replace('\xa0&\xa0'," ;
")[item.find('span',{'class':"_2_R_DZ"}).text.replace('\xa0&\xa0'," ; ").
```

Total data taken on **"Rating.xlsx"**

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  After putting data on excel, I import data by using

```
df = pd.DataFrame(pd.read_excel("Rating.xlsx"))
df
```

  we have 25597 rows and 5 columns
  Target column is New Rating which is integer
- Testing of Identified Approaches (Algorithms)
  As the target value is integer we should use Regression technique, here we will check R2 score.
  R2 score is **the proportion of the variance in the dependent variable that is predictable from the independent variable(s)**.

So if it is 100%, the two variables are perfectly correlated, i.e., with no variance at all. **R2=1−sum squared regression (SSR)total sum of squares (SST),=1−∑(yi−^yi)2∑(yi−¯y)2**..https://media.geeksforgeeks.org/wp-content/uploads/LR-cost-function-1.jpg

Another method is Lasso and Ridge technique.

Linear model equation is :-y=w[0]*x[0]+w[1]*x[1]+…..w[n]+x[n]+b

Here n is number of variable w is slope and b is intercept y output and x input.

Linear regression looks for optimizing w and b such that it minimizes the cost function.

Ridge and Lasso regression are **some of the simple techniques to reduce model complexity and prevent over-fitting** which may result from simple linear regression . Ridge Regression : In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients.

In ridge regression, however, the formula for the hat matrix should include the regularization penalty: $\mathbf{H_{ridge} = X(X'X + \lambda I)^{-1}X}$, which gives $df_{ridge} = trH_{ridge}$, which is no longer equal to m. Some ridge regression software produce information criteria based on the OLS formula.

Lasso regression is a type of linear regression that **uses shrinkage**. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). ... The acronym “LASSO” stands for Least Absolute Shrinkage and Selection Operator.

I got all those things sklearn.

```
print ("total sum of squares", y)
print ("total sum of residuals ", g)
print ("r2 calculated", 1 - (g / y))
```

```
print("R2 score : %.2f" % r2_score(ytest,preds))
```

Variance:- In terms of linear regression, **variance** is a measure of how far observed values differ from the average of predicted values, i.e., their difference from the **predicted value mean**.

np.var(err), where **err** is an array of the differences between observed and predicted values and **np.var()** is the numpy array variance function.

Mean square error (MSE) is the average of the square of the errors. The larger the number the larger the error. **Error** in this case means the difference between the observed values y1, y2, y3, … and the predicted ones pred(y1), pred(y2), pred(y3), … We square each difference (pred(yn) – yn)) ** 2 so that negative and positive values do not cancel each other out.

After these I used SVR and Random Forest technique

Given training vectors $x_i \in R^p$, $i=1,\ldots, n$, and a vector $y \in R^n$ ε-SVR solves the following primal problem:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$
$$\text{subject to } y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i, \quad w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*, \quad \zeta_i, \zeta_i^* \geq 0, \quad i=1,\ldots,n$$

Here, we are penalizing samples whose prediction is at least ε away from their true target. These samples penalize the objective by ζ_i or ζ_i^*, depending on whether their predictions lie above or below the ε tube.

The dual problem is

$$\min_{\alpha,\alpha^*} \frac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \varepsilon e^T (\alpha + \alpha^*) - y^T (\alpha - \alpha^*) \quad \text{subject to } e^T (\alpha - \alpha^*) = 0 \quad 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i=1,\ldots,n$$

where e is the vector of all ones, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.

The prediction is:

$$\sum_{i \in SV} (\alpha_i - \alpha_i^*) K(x_i, x) + b$$

These parameters can be accessed through the attributes `dual_coef_` which holds the difference $\alpha_i - \alpha_i^*$, `support_vectors_` which holds the support vectors, and `intercept_` which holds the independent term b

. LinearSVR

The primal problem can be equivalently formulated as

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \max(0, |y_i - (w^T \phi(x_i) + b)| - \varepsilon),$$

where we make use of the epsilon-insensitive loss, i.e. errors of less than ε are ignored. This is the form that is directly optimized by **`LinearSVR`**.

A random forest regressor.

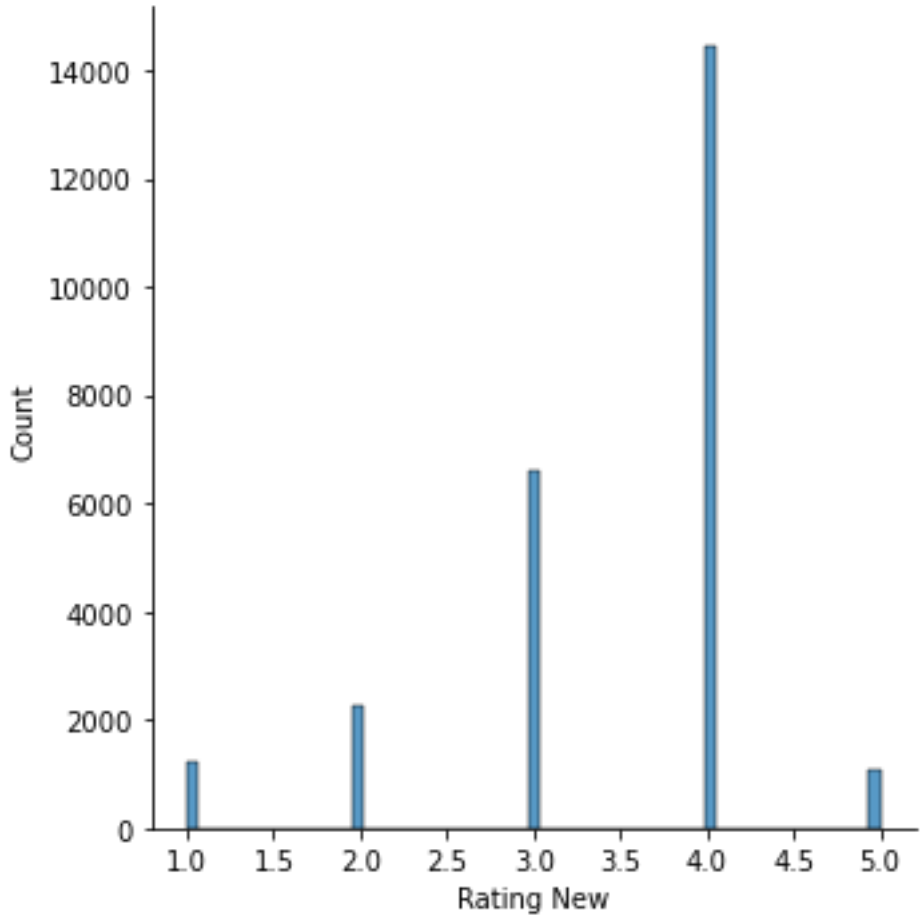
A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

```
from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import RandomForestRegressor
```

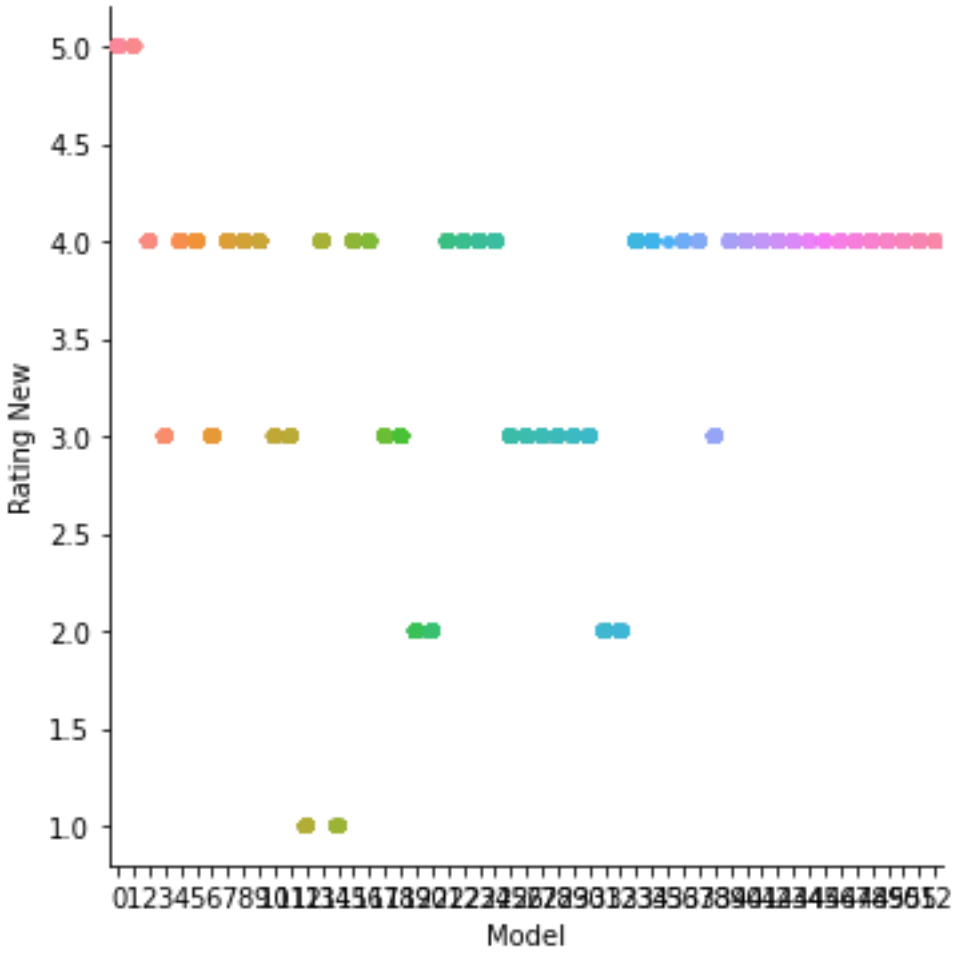
```
parameters = {'criterion':['mse', 'mae'],'max_features':["auto", "sqrt", "log2"]}
rf =RandomForestRegressor()
clf = GridSearchCV(rf,parameters)
clf.fit(xtrain,ytrain)

print(clf.best_params_)
rf= RandomForestRegressor(criterion="mse",max_features="auto")
rf.fit(xtrain, ytrain)
rf.score(xtrain, ytrain)
pred_decision = rf.predict(xtest)

rfs = r2_score(ytest,pred_decision)
print('R2 Score:',rfs*100)

rfscore = cross_val_score(rf,features,target,cv=5)
rfc = rfscore.mean()
print('Cross Val Score:',rfc*100)
```

- Visualizations

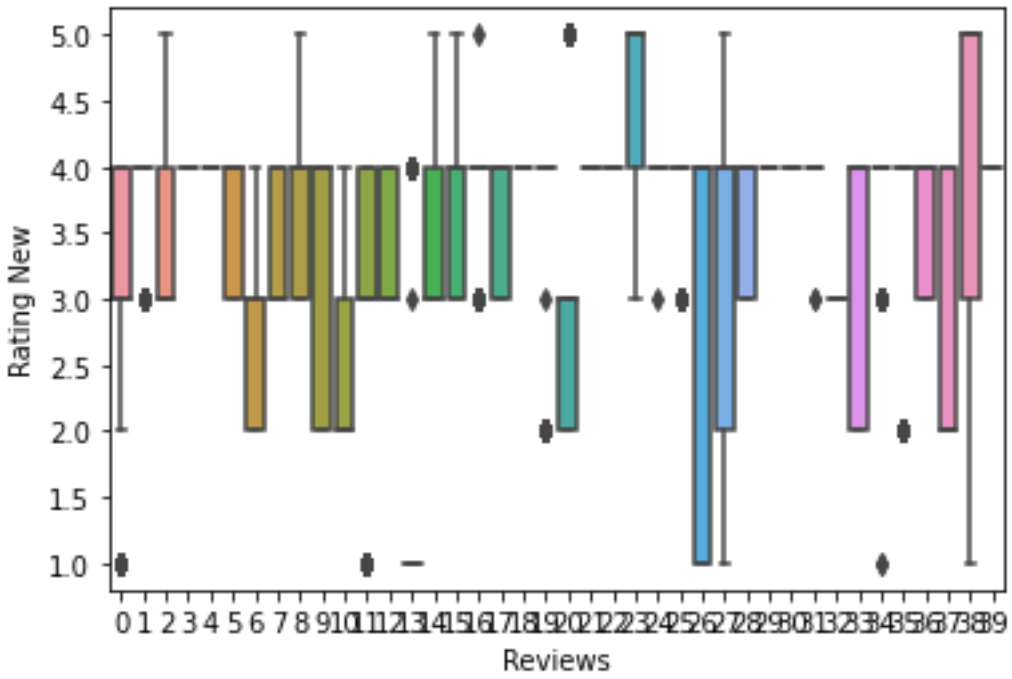
```
for i in df.columns:
  sns.displot(df[i])
```




```
sns.catplot(x="Model", y="Rating New", data=df)
```




```
sns.boxplot(x="Reviews", y="Rating New", data=df)
```

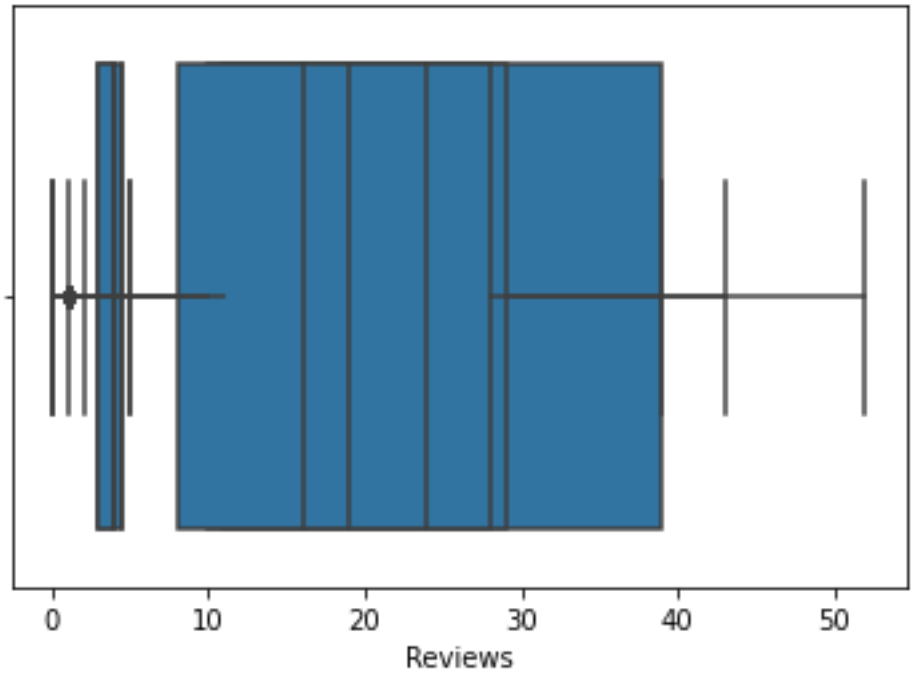


- Interpretation of the Results

In the graph we can see that most of the rating coming 4.

As Model, Review, Num _of_Ratings is object we need to encode it to int, for that I take LabelEncoder from sk learn.

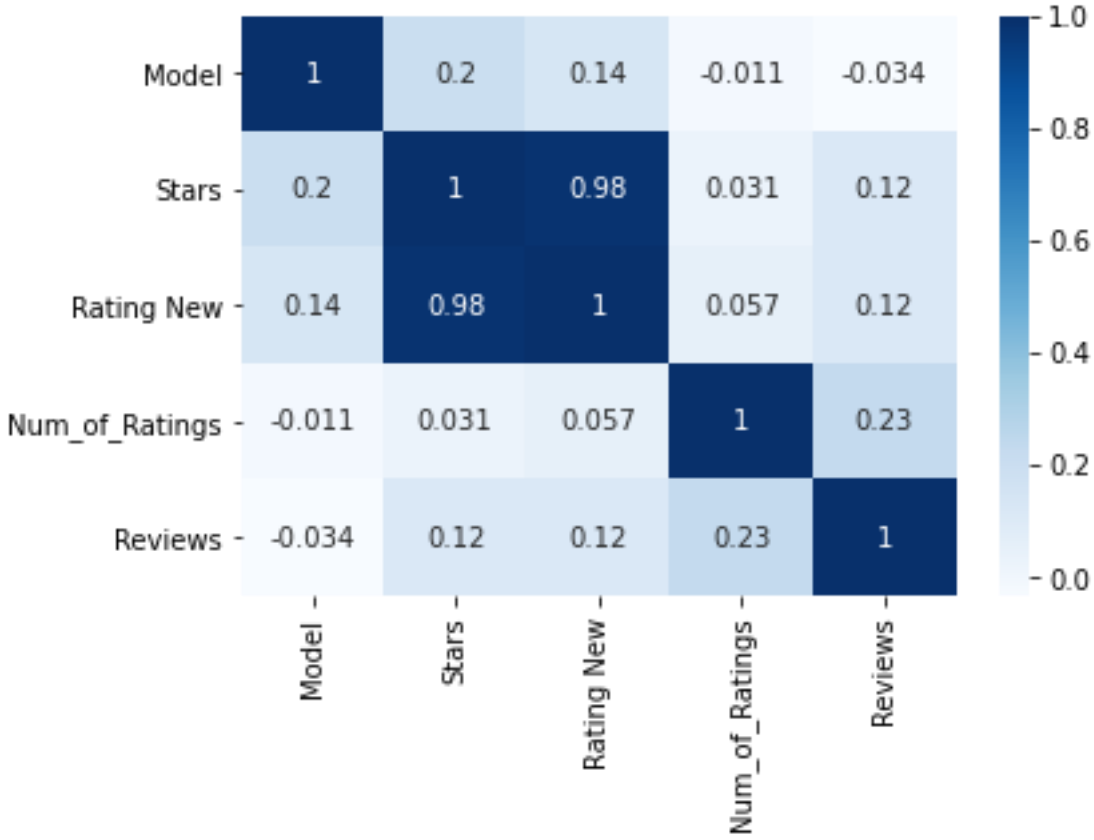
```
import sklearn
from sklearn.preprocessing import LabelEncoder
lencode=LabelEncoder()
df['Model']=lencode.fit_transform(df['Model'])
df['Num_of_Ratings']=lencode.fit_transform(df['Num_of_Ratings'])
df['Reviews']=lencode.fit_transform(df['Reviews'])
```



	Model	Stars	Rating New	Num_of_Ratings	Reviews
Model	1.000000	0.202469	0.139641	-0.010807	-0.033939
Stars	0.202469	1.000000	0.984859	0.030515	0.118098
Rating New	0.139641	0.984859	1.000000	0.056753	0.116174
Num_of_Ratings	-0.010807	0.030515	0.056753	1.000000	0.231049
Reviews	-0.033939	0.118098	0.116174	0.231049	1.000000

After consider corelation we can check there is no outlier.

Heatmap



CONCLUSION

- Key Findings and Conclusions of the Study

  R2 score of linear regression is

```
R2 score:: 0.9745942510343873
mean_squared_error is
error:
0.02028275074064433
```

R2 score of Lasso and Ridge

```
0.9743544142113998
0.9743544860114564
```

For ElasticNet

```
0.9743544561978105
```

For svr technique:- kernellist=['linear','poly','rbf']

```
0.9685977351545538
0.9388288211600813
0.9932701928285411
```

After ensemble process

```
R2 Score: 100.0
Cross Val Score: 100.0
Mean absolute error:: 0.11410932460629734
Mean squared error:: 0.02028275074064433
Root mean square:: 0.1424175225898988
```

Model:-

```
loaded_model = pickle.load(open('rating.pkl', 'rb'))
result = loaded_model.score(xtest, ytest)
print(result)
```

```
1.0
```

```
conclusion=pd.DataFrame([loaded_model.predict(xtest)[:],pred_decision[:]],index=["Predicted","Orginal"])
```

	0	1	2	3	4	5	6	7	8	9	...	7670	7671	7672	7673	7674	7675	7676	7677	7678	7679
Predicted	4.0	4.0	3.0	2.0	4.0	4.0	1.0	4.0	3.0	3.0	...	2.0	5.0	3.0	4.0	2.0	3.0	3.0	4.0	5.0	2.0
Orginal	4.0	4.0	3.0	2.0	4.0	4.0	1.0	4.0	3.0	3.0	...	2.0	5.0	3.0	4.0	2.0	3.0	3.0	4.0	5.0	2.0

2 rows × 7680 columns

- Learning Outcomes of the Study in respect of Data Science

I had total 25597 data and I use 7680 for testing purpose , result is show identical I got 100% results. It might differ if I not done ensamble process, but model is ok to go.

- Limitations of this work and Scope for Future Work

As per the requirement I adjust the decimal value, so if I did not do that might be results little differ.

No all the customer rate the product so it does not mean data is exactly correct.

I think if we got proper rating from, we can improve this.

Thank you

Total code you can get from my github
https://github.com/bishwa2017/Rating