```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
train_set = pd.read_csv('/content/drive/MyDrive/RF/train.csv')
print(train_set.shape)
train_set.head(3)
```

```
    (15730, 16)
```

	id	title	Rating	maincateg	platform	price1	actprice1	Offer %	norati
0	16695	Fashionable & Comfortable Bellies For Women (...	3.9	Women	Flipkart	698	999	30.13%	3
1	5120	Combo Pack of 4 Casual Shoes Sneakers For Men ...	3.8	Men	Flipkart	999	1999	50.03%	53
2	18391	Cilia Mode Leo Sneakers For Women (White)	4.4	Women	Flipkart	2749	4999	45.01%	1

```
# Loading X_train & y_train

X_train_orig = train_set.drop(['Offer %', 'price1'], axis=1)
print(X_train_orig.shape)  # same as X_test !
X_train_orig.head(2)
```

```
    (15730, 14)
```

	id	title	Rating	maincateg	platform	actprice1	norating1	noreviews1	star_5f	star_4f	star_3f	st
0	16695	Fashionable & Comfortable Bellies For Women (...	3.9	Women	Flipkart	999	38.0	7.0	17.0	9.0	6.0	
1	5120	Combo Pack of 4 Casual Shoes Sneakers For Men ...	3.8	Men	Flipkart	1999	531.0	69.0	264.0	92.0	73.0	

```
# y_train
y_train_offer = train_set['Offer %']
y_train_price = train_set['price1']
print(y_train_offer.head())
print(y_train_price.head())
y_train_price.shape
```

```
0    30.13%
1    50.03%
2    45.01%
3    15.85%
4    40.02%
Name: Offer %, dtype: object
0     698
1     999
2    2749
3     518
4    1379
Name: price1, dtype: int64
(15730,)
```

```
X_train_orig.isna().sum()
```

```
id              0
title           0
Rating          0
maincateg     526
platform        0
actprice1       0
norating1     678
noreviews1    578
star_5f       588
star_4f       539
star_3f       231
star_2f         0
star_1f         0
fulfilled1      0
dtype: int64
```

```
# Filling maincateg NaN using title
def fill_maincateg(df):
    for ind, item in enumerate(df.maincateg):
        # print(item)

        # how else to check if item is nan
       if(item!="Men" and item != "Women"):
            # print(df.title[ind])
            #if(df.title[ind].str.contains('Men')):
            if("Men" in df.title[ind]):
                df.loc[ind, "maincateg"] = 'Men'
            else:
                df.loc[ind, "maincateg"] = 'Women'
    print("Done")

    return df
```

```
train_na_cols = {'norating1': X_train_orig.norating1.mean(), 'noreviews1': X_train_orig.noreviews1.mean()}
train_na_cols
```

```
{'norating1': 3057.6607759766143, 'noreviews1': 423.97630675818374}
```

```
# for encoding 'train_set'

def encode_train_cols(X):
    # Filling maincateg using title
    #X.maincateg = X.maincateg.fillna('Men' if X.title.str.con)
    fill_maincateg(X)

    # Drop "title" & "id" & ratings
    cols_to_drop = ['id', 'title', 'star_5f', 'star_4f', 'star_3f', 'star_2f', 'star_1f']
    X.drop(cols_to_drop, axis=1, inplace=True)

    # Handling Missing values
    # replacing with most common value in train set
    X.fillna(train_na_cols, inplace=True)
    # OHE "maincateg" & "platform"
    dummy_features = ['maincateg', 'platform']
    X = pd.get_dummies(X, columns=dummy_features)

    return X
```

```
test_na_cols = {'Rating': X_train_orig.Rating.mean()}
test_na_cols
```

```
{'Rating': 4.012873490146217}
```

```
# for encoding 'test set'
def encode_test_cols(X):
    # Filling maincateg using title
    fill_maincateg(X)

    # Drop "title" & "id" & 'norating1'
    cols_to_drop = ['id', 'title', 'star_5f', 'star_4f', 'star_3f', 'star_2f', 'star_1f']
    X.drop(cols_to_drop, axis=1, inplace=True)

    # Handling Missing values
    # replacing with most common value in train set
    X.fillna(test_na_cols, inplace=True)

    # OHE "maincateg" & "platform"
    dummy_features = ['maincateg', 'platform']
    X = pd.get_dummies(X, columns=dummy_features)

    return X
```

```
x_train = encode_train_cols(X_train_orig)
print(x_train.shape)
x_train.head()
```

```
Done
(15730, 9)
```

	Rating	actprice1	norating1	noreviews1	fulfilled1	maincateg_Men	maincateg_Women	platform_Amazon	platform_Flip
0	3.9	999	38.0	7.0	0	0	1	0	
1	3.8	1999	531.0	69.0	1	1	0	0	
2	4.4	4999	17.0	4.0	1	0	1	0	
3	4.2	724	46413.0	6229.0	1	1	0	0	
4	3.9	2299	77.0	3.0	1	1	0	0	

```
X_train_orig.head()

# cols dropped
# na filled
# ohe left
```

	Rating	maincateg	platform	actprice1	norating1	noreviews1	fulfilled1
0	3.9	Women	Flipkart	999	38.0	7.0	0

Training Model - RF

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
```

```
y_train_price.head()
```

```
0     698
1     999
2    2749
3     518
4    1379
Name: price1, dtype: int64
```

```
X_train, X_valid, y_train, y_valid = train_test_split(x_train,y_train_price,test_size=0.15, random_state=0)
print(X_train.shape)
print(X_valid.shape)
```

```
(13370, 9)
(2360, 9)
```

```
rf = RandomForestRegressor(n_estimators=20)
rf.fit(X_train, y_train)
```

```
▾         RandomForestRegressor
RandomForestRegressor(n_estimators=20)
```

```
print(rf.score(X_train, y_train))
rf.score(X_valid, y_valid)

# very less on valid - overfit
```

```
0.981054846889404
0.9048361730765091
```

```
from sklearn.metrics import mean_squared_error

pred_train = rf.predict(X_train)
print("Train: ", np.sqrt(mean_squared_error(y_train, pred_train)))

pred_val = rf.predict(X_valid)
print("Val: ", np.sqrt(mean_squared_error(y_valid, pred_val)))
```

```
Train:  89.6760726719664
Val:  196.53515735512016
```

Generate submission file for rf

//////////////////////////

```
X_test = pd.read_csv('/content/drive/MyDrive/RF/test.csv')
test_id = X_test['id']
print(test_id[:3])
```

```
0     2242
1    20532
```

```
2    10648
Name: id, dtype: int64
```

```
X_test = encode_test_cols(X_test)
X_test.head()
```

```
Done
```

	Rating	actprice1	norating1	noreviews1	fulfilled1	maincateg_Men	maincateg_Women	platform_Amazon	platform_Flip
0	3.8	999	27928	3543	1	1	0	0	
1	3.9	499	3015	404	1	0	1	0	
2	3.9	999	449	52	1	0	1	0	
3	3.9	2999	290	40	1	1	0	0	
4	3.9	999	2423	326	0	1	0	0	

```
pred_test = rf.predict(X_test)
pred_test[:5]
```

```
array([422.27818182, 291.34960317, 456.        , 902.8       ,
       399.7       ])
```

```
subm_file = pd.DataFrame(test_id)
subm_file['price1'] = pred_test
subm_file.head()
```

	id	price1
0	2242	422.278182
1	20532	291.349603
2	10648	456.000000
3	20677	902.800000
4	12593	399.700000

```
subm_file.to_csv('5_rf.csv', index=False)
```

Score: 199 Damn??

function for fitting RF model

```
def score(model, title):
    model.fit(X_train, y_train)

    print("RMSE for", title, ": ")

    pred_train = model.predict(X_train)
    print("Train: ", np.sqrt(mean_squared_error(y_train, pred_train)))

    pred_val = model.predict(X_valid)
    print("Val: ", np.sqrt(mean_squared_error(y_valid, pred_val)))

    # print("Accurancy for ", title)
    # print("\tTrain: ", model.score(X_train, y_train))
    # print("\tTest: ", model.score(X_valid, y_valid))
```

```
RF = RandomForestRegressor(n_estimators=1000, max_depth=10, random_state=0)
```

```
score(RF, "RandomForest")
```

```
RMSE for RandomForest :
Train:  173.29092056470597
Val:  220.28506755619608
```

Generate submission file for RF

////////////////////////////

```
pred_test2 = RF.predict(X_test)
```

```
subm_file = pd.DataFrame(test_id)
subm_file['price1'] = pred_test2
subm_file.head()
```

	id	price1
0	2242	436.641355
1	20532	293.234288
2	10648	443.537603
3	20677	950.391864
4	12593	412.550063

```
subm_file.to_csv('5_rf_2.csv', index=False)
```

Score: 225 hmm, expected.

///

My RF

```
SRF = RandomForestRegressor(max_depth=30,max_features=5,min_samples_leaf=1,min_samples_split=2,n_estimators=580,bootstrap=Tr
score(SRF, "SRF")
```

```
RMSE for SRF :
Train:  80.61050858818628
Val:  183.9702786419364
```

```
def gen_subm_file(model):
    X_test = pd.read_csv('/content/drive/MyDrive/RF/test.csv')
    test_id = X_test['id']

    X_test = encode_test_cols(X_test)

    pred_test = model.predict(X_test)

    subm_file = pd.DataFrame(test_id)
    subm_file['price1'] = pred_test

    return subm_file
```

```
subm_file = gen_subm_file(SRF)
subm_file.to_csv("5_rf_3.csv", index=False)
```

```
Done
```

```
subm_file = pd.read_csv('5_rf_3.csv')
print(subm_file.isna().sum())
subm_file.head()
```

```
id        0
price1    0
dtype: int64
```

	id	price1
0	2242	427.128965
1	20532	294.739996
2	10648	451.039178
3	20677	991.706897
4	12593	399.716476

Score: 191 best

Feature Scaling

```
x_train.shape
```

```
(15730, 9)
```

```
def normalize(X):
    features = X.columns
    X[features] /= X_train[features].max()
    return X
```

```
X_train_norm = normalize(x_train.copy())
X_train_norm.head()
```

	Rating	actprice1	norating1	noreviews1	fulfilled1	maincateg_Men	maincateg_Women	platform_Amazon	platform_Flip
0	0.78	0.074005	0.000131	0.000154	0.0	0.0	1.0	0.0	
1	0.76	0.148085	0.001831	0.001518	1.0	1.0	0.0	0.0	
2	0.88	0.370324	0.000059	0.000088	1.0	0.0	1.0	0.0	
3	0.84	0.053634	0.160060	0.137058	1.0	1.0	0.0	0.0	
4	0.78	0.170309	0.000266	0.000066	1.0	1.0	0.0	0.0	

```
x_train.head()  # should not get normalized
```

	Rating	actprice1	norating1	noreviews1	fulfilled1	maincateg_Men	maincateg_Women	platform_Amazon	platform_Flip
0	3.9	999	38.0	7.0	0	0	1	0	
1	3.8	1999	531.0	69.0	1	1	0	0	
2	4.4	4999	17.0	4.0	1	0	1	0	
3	4.2	724	46413.0	6229.0	1	1	0	0	
4	3.9	2299	77.0	3.0	1	1	0	0	

```
X_train_norm.shape  # should be (15730, 9)
```

```
(15730, 9)
```

```
y_train_offer = y_train_offer.str.replace(r'%', '')
y_train_offer = y_train_offer.astype(float)
y_train_offer.head()
```

```
0    30.13
1    50.03
2    45.01
3    15.85
4    40.02
Name: Offer %, dtype: float64
```

```
y_train_offer /= 100
y_train_offer.head()
y_train_offer.describe()
```

```
count    15730.000000
mean         0.468025
std          0.192687
min          0.000000
25%          0.359400
50%          0.500700
75%          0.601600
max          0.889300
Name: Offer %, dtype: float64
```

```
y_train_offer.shape  # should be (15730,)
```

```
(15730,)
```

Training - after feature scaling

```
X_train2, X_valid2, y_train2, y_valid2 = train_test_split(X_train_norm,y_train_offer,test_size=0.15, random_state=0)
X_train2.shape
```

```
(13370, 9)
```

```
def score2(model, title):
    print("fitting the model..")
    model.fit(X_train2, y_train2)

    print("RMSE for", title, ": ")

    pred_train = model.predict(X_train2)
    print("Train: ", np.sqrt(mean_squared_error(y_train2, pred_train)))

    pred_val = model.predict(X_valid2)
    print("Val: ", np.sqrt(mean_squared_error(y_valid2, pred_val)))
```

```
rf2 = RandomForestRegressor(n_estimators=20)
score2(rf2, "RF2")
```

```
fitting the model..
RMSE for RF2 :
Train:  0.04771742652722249
Val:  0.11167064013664499
```

```
# offer is offer%
def predict_price(offer, test_actprice):
    # offer *= 100
    test_actprice -= (test_actprice * offer)
    return test_actprice


def gen_subm_file2(model):
    X_test = pd.read_csv('/content/drive/MyDrive/RF/test.csv')
    test_id = X_test['id']
    test_actprice = X_test['actprice1']

    X_test = encode_test_cols(X_test)
    X_test = normalize(X_test)

    pred_test_offer = model.predict(X_test)
#     print("offer: ", pred_test_offer[:5])
    pred_test_price = predict_price(pred_test_offer, test_actprice)
#     print("price: ", pred_test_price[:5])

    subm_file = pd.DataFrame(test_id)
    subm_file['price1'] = pred_test_price

    return subm_file


subm_file = gen_subm_file2(rf2)
subm_file.to_csv("5_rf_norm.csv", index=False)
```

```
     Done
```

```
subm_file = pd.read_csv('5_rf_norm.csv')
subm_file.head()
```

	id	price1
0	2242	429.280290
1	20532	292.056009
2	10648	421.408170
3	20677	792.440765
4	12593	397.759937

```
# This is formatted as code
```

Score: 197 Why was expecting a significant inc after norm? ;-;

Norm Train using GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
rfc=RandomForestRegressor(random_state=0)
rfc.fit(X_train, y_train)

param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8,9,10],
    'criterion' :['squared_error']
}
param_grid
```

```
     {'n_estimators': [200, 500],
      'max_features': ['auto', 'sqrt', 'log2'],
         'max_depth': [4, 5, 6, 7, 8, 9, 10],
         'criterion': ['squared_error']}

# param_grid = {  'bootstrap': [True], 'max_depth': [5, 10, None], 'max_features': ['auto', 'log2'], 'n_estimators': [5, 6,


import warnings
warnings.filterwarnings('ignore')



CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
score2(CV_rfc, "CV RF")
```

```
fitting the model..
RMSE for CV RF :
Train:  0.12391851669903312
Val:  0.13590428983143404
```

```
CV_rfc.best_params_
```

```
{'criterion': 'squared_error',
 'max_depth': 10,
 'max_features': 'auto',
 'n_estimators': 500}
```

```
subm_file = gen_subm_file2(rf2)
subm_file.to_csv("5_rf_CV.csv", index=False)
```

```
Done
```

```
subm_file.head()
```

	id	price1
0	2242	429.280290
1	20532	292.056009
2	10648	421.408170
3	20677	792.440765
4	12593	397.759937

Score: 197

1m 50s completed at 7:14 PM