Mobile Price Classification

Hailah Alharthi

١٤/٠٧/٤٢

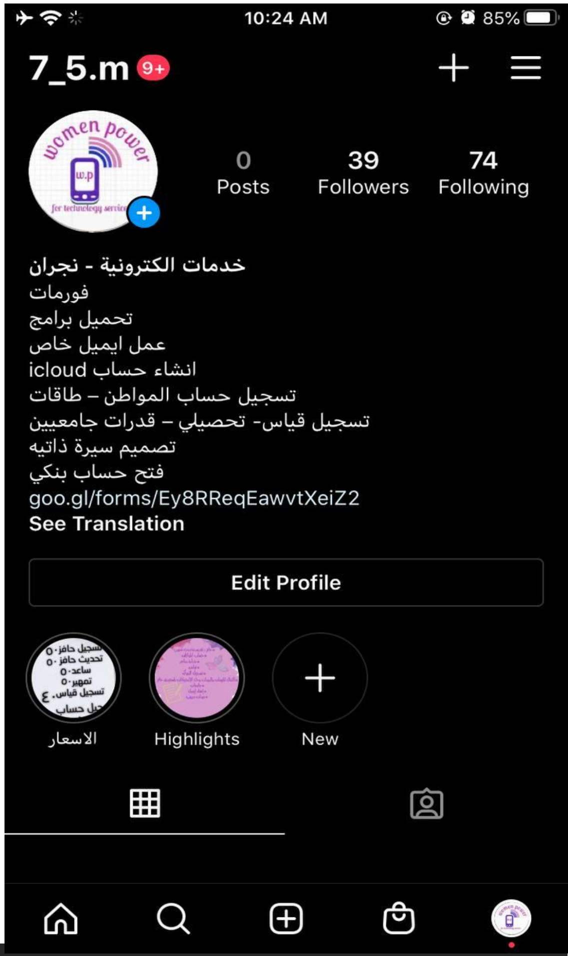
Outline

Problem Statement

I am always thinking about startup business, for example, if I were to open a business, how could I deal with that and what I need to know before I begin ?

Technology and electronics are my passion, in 2019 I started a small business that offered electronic services with some HW/SW Support in one of Najran university's bazaars. During that time I faced many challenges and broke many barriers. The business started with success, but the bazaars had a limited duration, and finding high-quality phone pieces was difficult because of my ignorance of the various suppliers; I realize now that I needed to study the market first.

Searching for a dataset related to my business, I found Mr. Bob on Kagel asking for users to Classify the price ranges of an array of devices based on various features. With the skills I have acquired over the past 14 weeks, I am beginning to solve this problem with Data Science.



خدمات الكترونية - نجران
فورمات
تحميل برامج
عمل ايميل خاص
icloud انشاء حساب
تسجيل حساب المواطن – طاقات
تسجيل قياس- تحصيلي – قدرات جامعيين
تصميم سيرة ذاتيه
فتح حساب بنكي
goo.gl/forms/Ey8RReqEawvtXeiZ2
See Translation

What is the data ?

In this project, we obtain to explore and analyze a dataset that hold specifications of 2000 mobile phones as well as attempt to predict best price ranges for a list of mobile phones in the market by applying various machine learning algorithm.

Target :

Our Target is price range , we have four range [0 , 1, 2, 3]

The target variable indicates as below:

- 0 (low cost)
- 1 (medium cost)
- 2 (high cost)
- 3 (very high cost)

The problem can be solved as classification problem. Since there are four discrete classes.

Work stream

Data Collection

Google

Kaggle

Exploratory Data Analysis

Loading

Processing

Visualizing

Defined Models

Baseline

Random forest

K nearest neighbor

Decision tree

Stacking

Validate the best model

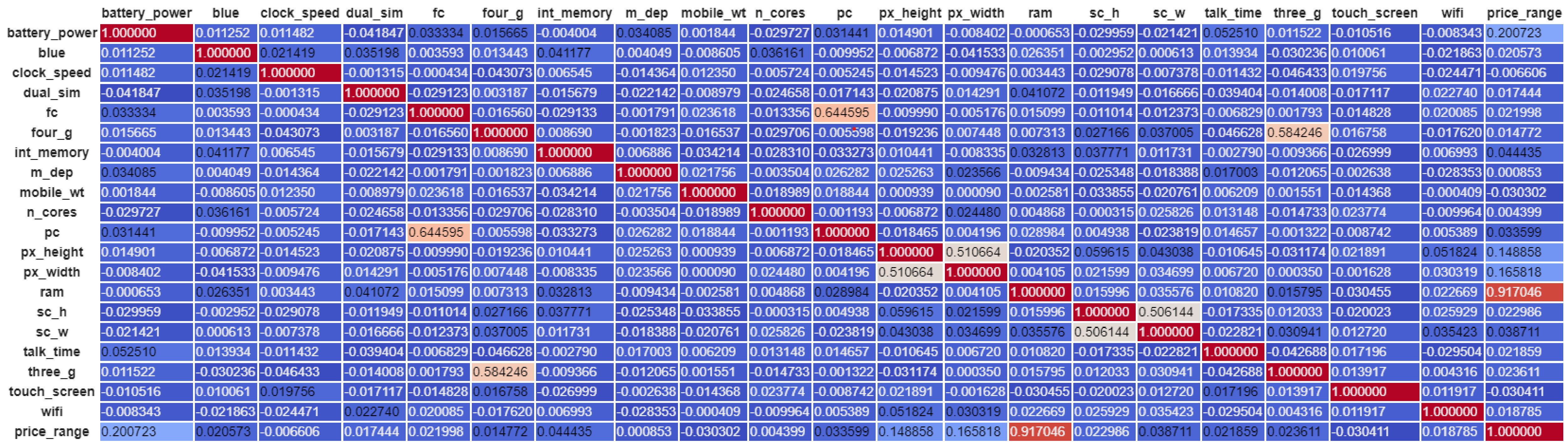
cross validation

Comparing models

boxplot

Optimize the Model

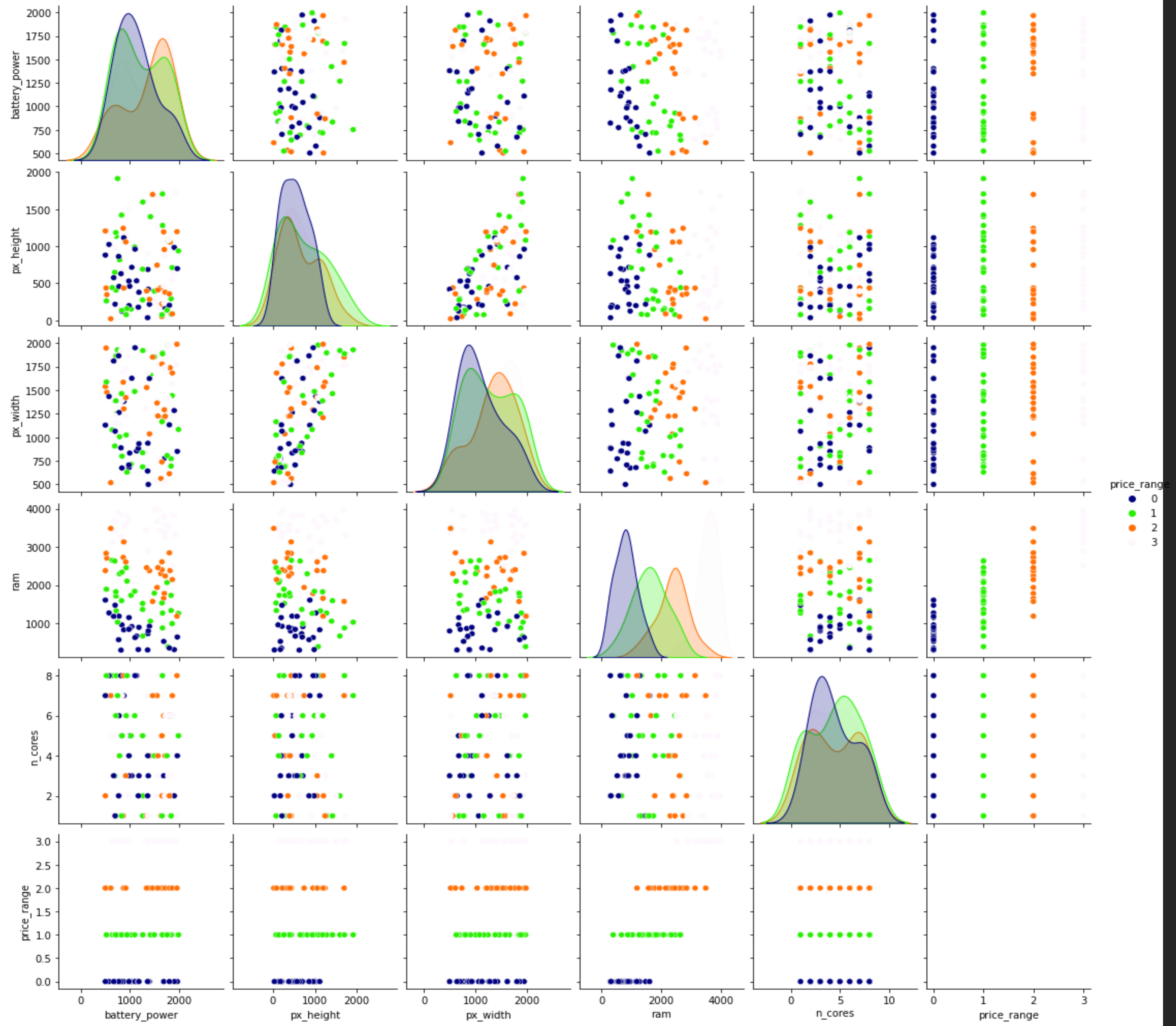
Grid Search



	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
battery_power	1.000000	0.011252	0.011482	-0.041847	0.033334	0.015665	-0.004004	0.034085	0.001844	-0.029727	0.031441	0.014901	-0.008402	-0.000653	-0.029959	-0.021421	0.052510	0.011522	-0.010516	-0.008343	0.200723
blue	0.011252	1.000000	0.021419	0.035198	0.003593	0.013443	0.041177	0.004049	-0.008605	0.036161	-0.009952	-0.006872	-0.041533	0.026351	-0.002952	0.000613	0.013934	-0.030236	0.010061	-0.021863	0.020573
clock_speed	0.011482	0.021419	1.000000	-0.001315	-0.000434	-0.043073	0.006545	-0.014364	0.012350	-0.005724	-0.005245	-0.014523	-0.009476	0.003443	-0.029078	-0.007378	-0.011432	-0.046433	0.019756	-0.024471	-0.006606
dual_sim	-0.041847	0.035198	-0.001315	1.000000	-0.029123	0.003187	-0.015679	-0.022142	-0.008979	-0.024658	-0.017143	-0.020875	0.014291	0.041072	-0.011949	-0.016666	-0.039404	-0.014008	-0.017117	0.022740	0.017444
fc	0.033334	0.003593	-0.000434	-0.029123	1.000000	-0.016560	-0.029133	-0.001791	0.023618	-0.013356	0.644595	-0.009990	-0.005176	0.015099	-0.011014	-0.012373	-0.006829	0.001793	-0.014828	0.020085	0.021998
four_g	0.015665	0.013443	-0.043073	0.003187	-0.016560	1.000000	0.008690	-0.001823	-0.016537	-0.029706	-0.005598	-0.019236	0.007448	0.007313	0.027166	0.037005	-0.046628	0.584246	0.016758	-0.017620	0.014772
int_memory	-0.004004	0.041177	0.006545	-0.015679	-0.029133	0.008690	1.000000	0.006886	-0.034214	-0.028310	-0.033273	0.010441	-0.008335	0.032813	0.037771	0.011731	-0.002790	-0.009366	-0.026999	0.006993	0.044435
m_dep	0.034085	0.004049	-0.014364	-0.022142	-0.001791	-0.001823	0.006886	1.000000	0.021756	-0.003504	0.026282	0.025263	0.023566	-0.009434	-0.025348	-0.018388	0.017003	-0.012065	-0.002638	-0.028353	0.000853
mobile_wt	0.001844	-0.008605	0.012350	-0.008979	0.023618	-0.016537	-0.034214	0.021756	1.000000	-0.018989	0.018844	0.000939	0.000090	-0.002581	-0.033855	-0.020761	0.006209	0.001551	-0.014368	-0.000409	-0.030302
n_cores	-0.029727	0.036161	-0.005724	-0.024658	-0.013356	-0.029706	-0.028310	-0.003504	-0.018989	1.000000	-0.001193	-0.006872	0.024480	0.004868	-0.000315	0.025826	0.013148	-0.014733	0.023774	-0.009964	0.004399
pc	0.031441	-0.009952	-0.005245	-0.017143	0.644595	-0.005598	-0.033273	0.026282	0.018844	-0.001193	1.000000	-0.018465	0.004196	0.028984	0.004938	-0.023819	0.014657	-0.001322	-0.008742	0.005389	0.033599
px_height	0.014901	-0.006872	-0.014523	-0.020875	-0.009990	-0.019236	0.010441	0.025263	0.000939	-0.006872	-0.018465	1.000000	0.510664	-0.020352	0.059615	0.043038	-0.010645	-0.031174	0.021891	0.051824	0.148858
px_width	-0.008402	-0.041533	-0.009476	0.014291	-0.005176	0.007448	-0.008335	0.023566	0.000090	0.024480	0.004196	0.510664	1.000000	0.004105	0.021599	0.034699	0.006720	0.000350	-0.001628	0.030319	0.165818
ram	-0.000653	0.026351	0.003443	0.041072	0.015099	0.007313	0.032813	-0.009434	-0.002581	0.004868	0.028984	-0.020352	0.004105	1.000000	0.015996	0.035576	0.010820	0.015795	-0.030455	0.022669	0.917046
sc_h	-0.029959	-0.002952	-0.029078	-0.011949	-0.011014	0.027166	0.037771	-0.025348	-0.033855	-0.000315	0.004938	0.059615	0.021599	0.015996	1.000000	0.506144	-0.017335	0.012033	-0.020023	0.025929	0.022986
sc_w	-0.021421	0.000613	-0.007378	-0.016666	-0.012373	0.037005	0.011731	-0.018388	-0.020761	0.025826	-0.023819	0.043038	0.034699	0.035576	0.506144	1.000000	-0.022821	0.030941	0.012720	0.035423	0.038711
talk_time	0.052510	0.013934	-0.011432	-0.039404	-0.006829	-0.046628	-0.002790	0.017003	0.006209	0.013148	0.014657	-0.010645	0.006720	0.010820	-0.017335	-0.022821	1.000000	-0.042688	0.017196	-0.029504	0.021859
three_g	0.011522	-0.030236	-0.046433	-0.014008	0.001793	0.584246	-0.009366	-0.012065	0.001551	-0.014733	-0.001322	-0.031174	0.000350	0.015795	0.012033	0.030941	-0.042688	1.000000	0.013917	0.004316	0.023611
touch_screen	-0.010516	0.010061	0.019756	-0.017117	-0.014828	0.016758	-0.026999	-0.002638	-0.014368	0.023774	-0.008742	0.021891	-0.001628	-0.030455	-0.020023	0.012720	0.017196	0.013917	1.000000	0.011917	-0.030411
wifi	-0.008343	-0.021863	-0.024471	0.022740	0.020085	-0.017620	0.006993	-0.028353	-0.000409	-0.009964	0.005389	0.051824	0.030319	0.022669	0.025929	0.035423	-0.029504	0.004316	0.011917	1.000000	0.018785
price_range	0.200723	0.020573	-0.006606	0.017444	0.021998	0.014772	0.044435	0.000853	-0.030302	0.004399	0.033599	0.148858	0.165818	0.917046	0.022986	0.038711	0.021859	0.023611	-0.030411	0.018785	1.000000

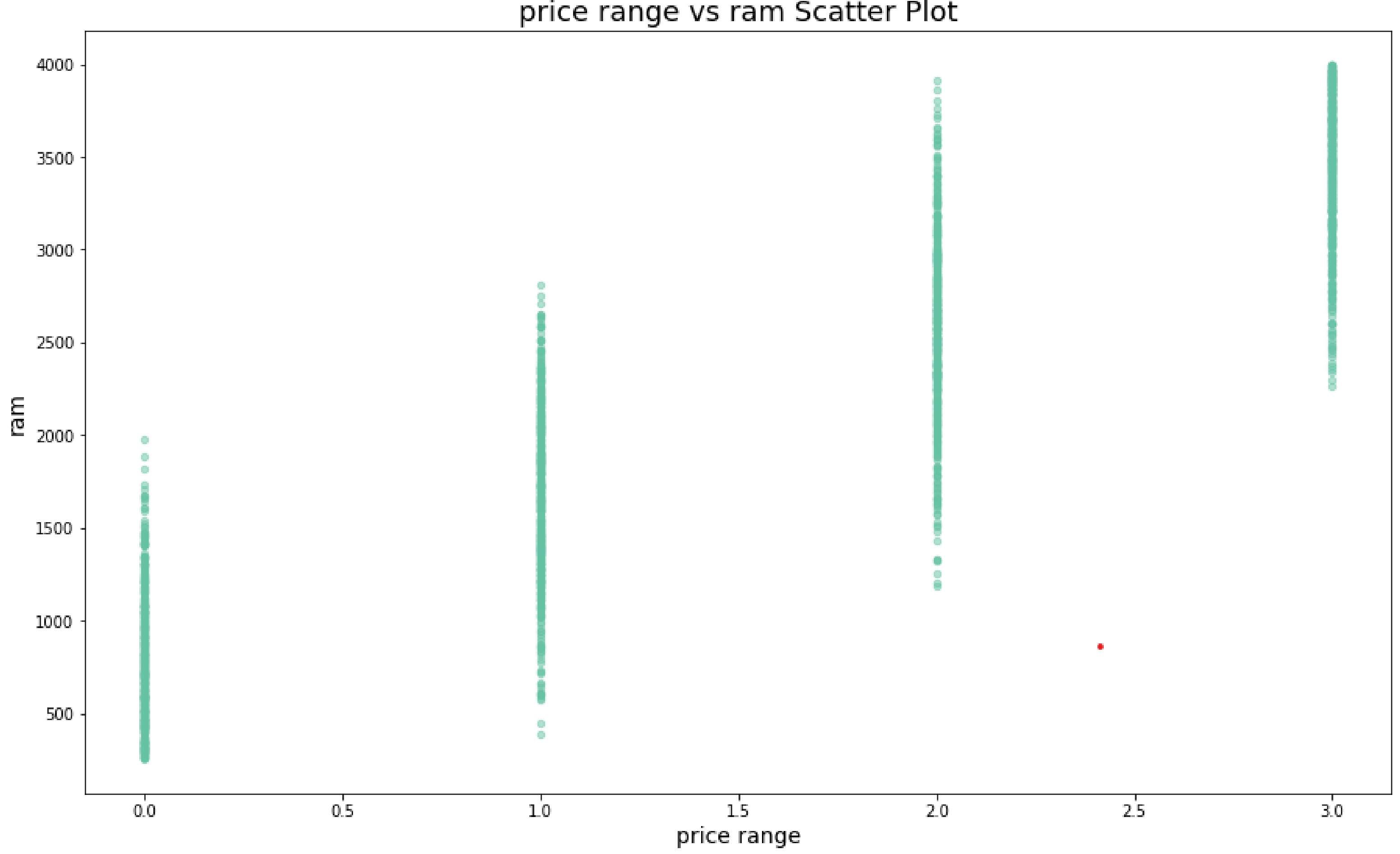
Heatmap

to understand the relationship between the feature.



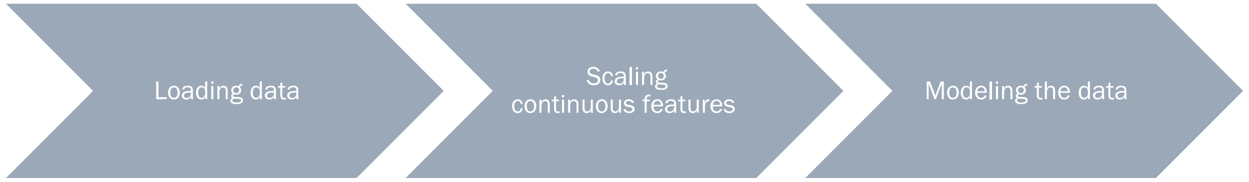
Pair plot

To explore how data change over deferent type of price range



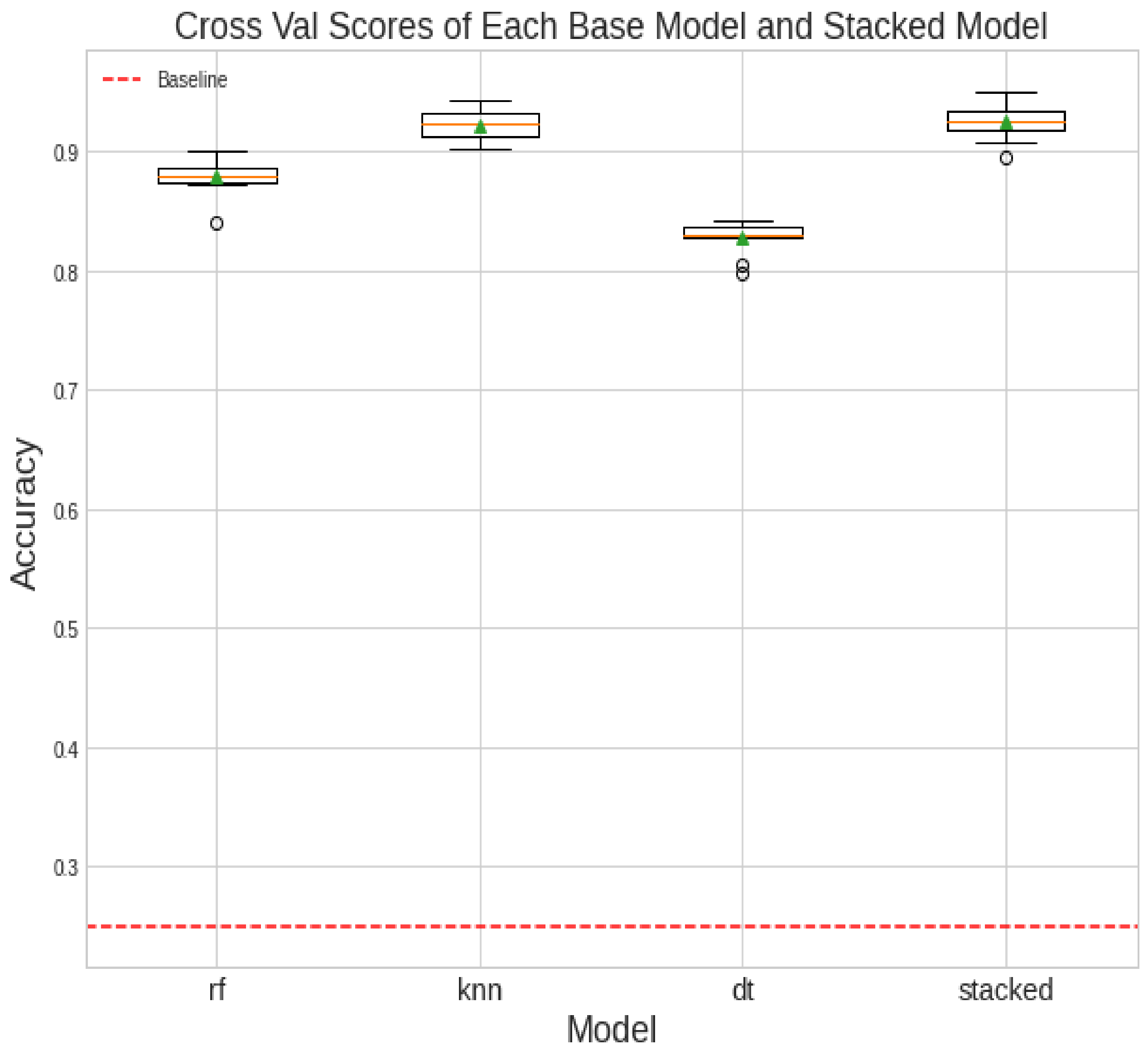
RAM VS Price Range

١٤/٠٧/٤٢



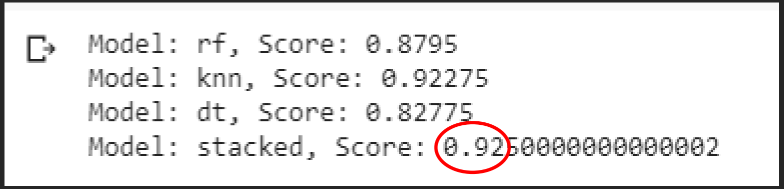
Machine learning

Using machine learning to build system can predict the range of price depend on what is the specification of mobile

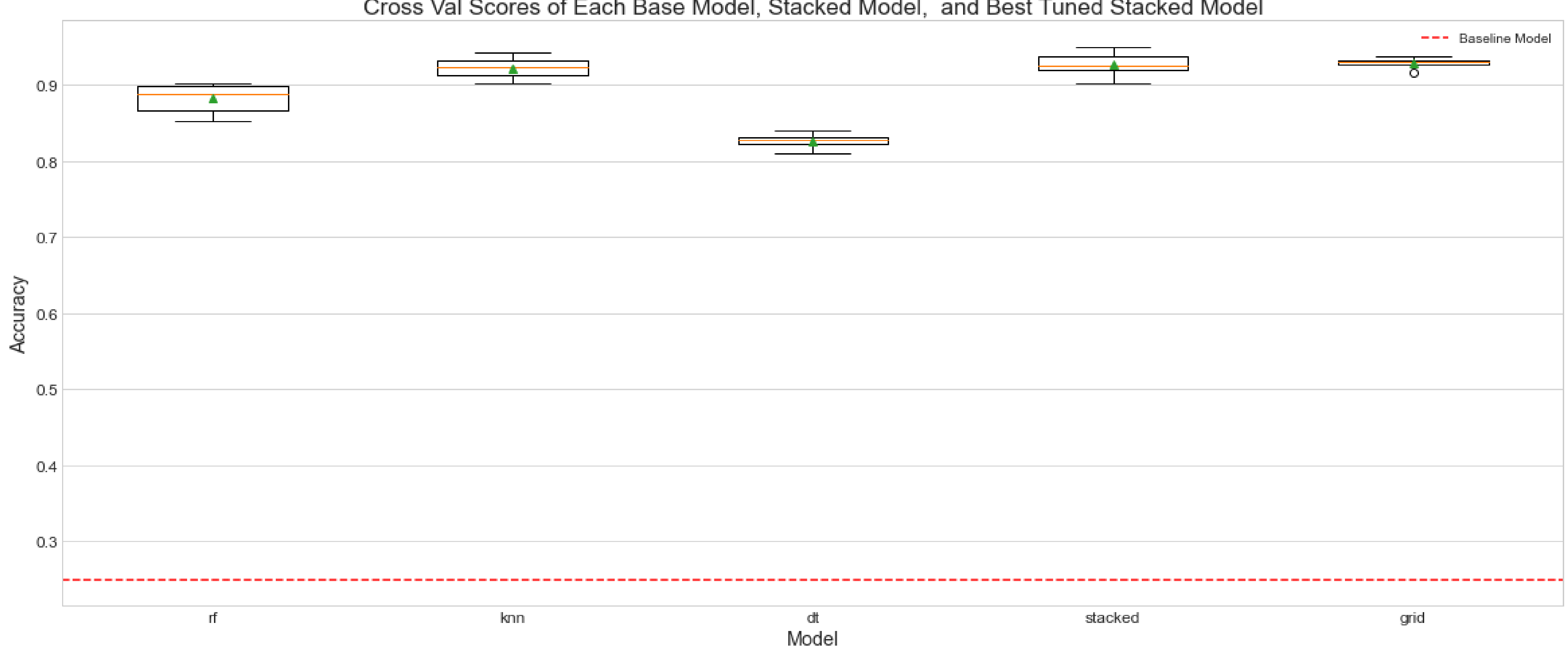


Applied Models

- ➢ Random forest model (rf)
- ➢ K nearest neighbor model (knn)
- ➢ Decision tree model (dt)
- ➢ Stacking model (staked)

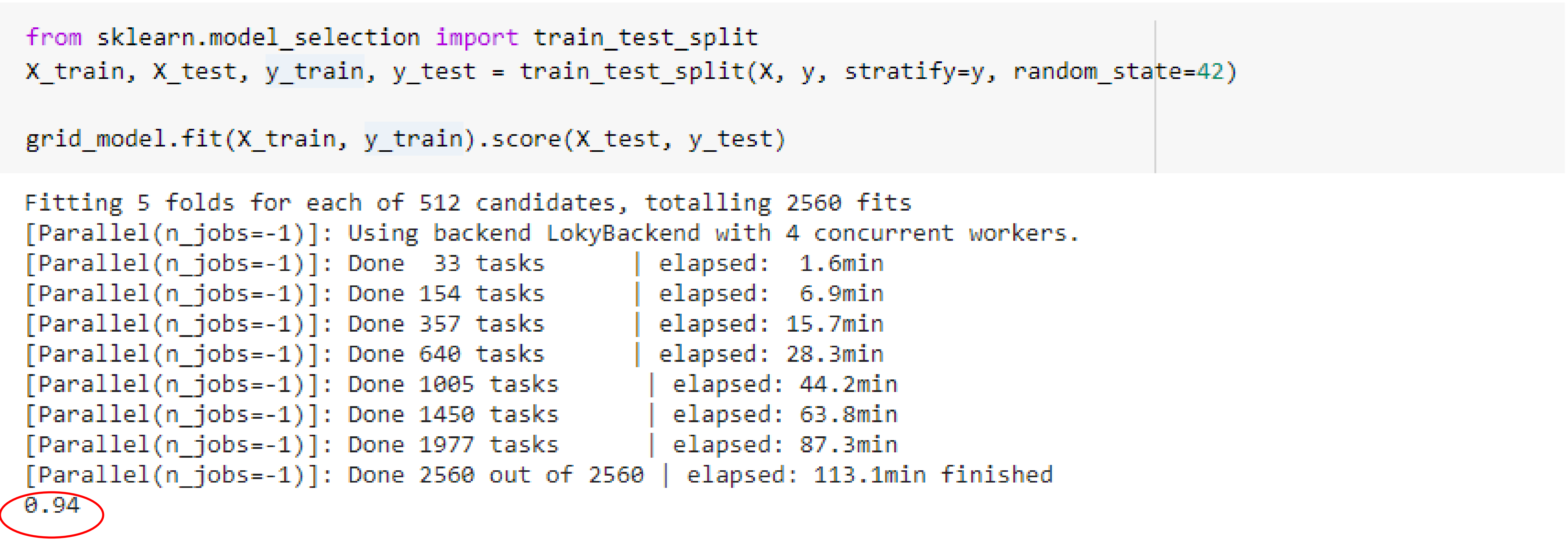


```
Model: rf, Score: 0.8795
Model: knn, Score: 0.92275
Model: dt, Score: 0.82775
Model: stacked, Score: 0.9250000000000002
```



Improve stacked Model

Optimizing score Using grid search



```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)

grid_model.fit(X_train, y_train).score(X_test, y_test)
```

```
Fitting 5 folds for each of 512 candidates, totalling 2560 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  6.9min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed: 15.7min
[Parallel(n_jobs=-1)]: Done 640 tasks      | elapsed: 28.3min
[Parallel(n_jobs=-1)]: Done 1005 tasks      | elapsed: 44.2min
[Parallel(n_jobs=-1)]: Done 1450 tasks      | elapsed: 63.8min
[Parallel(n_jobs=-1)]: Done 1977 tasks      | elapsed: 87.3min
[Parallel(n_jobs=-1)]: Done 2560 out of 2560 | elapsed: 113.1min finished
0.94
```

split the data and apply the stack model

The score is jump from 92% to 94% that means the stacking mode is doing good after optimizing with best parameter

Tech stack

Python	Pandas	Numpy
Seaborn	matplotlib	Scikitlearn

Future work

1

Add

- Add another model to stack that help to up the accuracy

2

Find

- Find dataset with modern specifications to Make predictions on new and valuable data .

3

Build

- Build website for predict the price depends on our model

data is the new oil
we need to find it, extract it, refine it, distribute it and monetize it.
David Buckingham

Thank you

Interested to hear your comments &question