MODEL COMPARISON ON MOBILE PRICE PREDICTION

Authors: Jingjie Ma; Ella Wang; Han Zhang
Data Science And Analyst Program, Georgetown University

INTRODUCTION

Mobile phones have been well developed for over 40 years in the market. As they were getting "evolved" in the gradually competitive market, developers started to add other technologies to the basic functions such as texting and calling. So nowadays, our phone is much lighter and more convenient compared to the past. These technologies and features also determine the high variance in mobile phone price.

Our dataset consists of different features of mobile phones. The goal is to compare different classification methods used in the prediction of the mobile price, and explore the algorithms and packages in Python.

DATASET OVERVIEW

This dataset gathered from Kaggle has 2000 observations and 21 variables in total. Since we are interested in predicting the price, we will use the price range as the response variable.

The response variable is preprocessed and is split into 4 classes. Thus, we are going to train multiclass classification models.

Predictors

- **battery_power**: Total energy a battery can store (quantitative)
- **clock_speed**: Speed at which microprocessor executes instructions (quantitative)
- **ram**: Random Access Memory in Megabytes (quantitative)
- **dual_sim**: Has dual sim support or not (0 for no, 1 for yes)
- **int_memory**: Internal Memory in Gigabytes (quantitative)
- **n_cores**: Number of cores of processor (quantitative)
- **four_g**: Has 4G or not (0 for no, 1 for yes)
- **three_g**: Has 3G or not (0 for no, 1 for yes)
- **blue**: Has bluetooth or not (0 for no, 1 for yes)
- **touch_screen**: Has touch screen or not (0 for no, 1 for yes)
- **wifi**: Has wifi or not (0 for no, 1 for yes)
- **fc**: Front Camera megapixels (quantitative)
- **pc**: Primary Camera megapixels (quantitative)
- **m_dep**: Mobile Depth in cm (quantitative)
- **mobile_wt**: Weight of mobile phone (quantitative)
- **px_height**: Pixel Resolution Height (quantitative)
- **px_width**: Pixel Resolution Width (quantitative)
- **sc_h**: Screen Height of mobile in cm (quantitative)
- **sc_w**: Screen Width of mobile in cm (quantitative)
- **talk_time**: Longest time that a single battery charge will last in hours (quantitative)

Response Variable

- **Price Range**: Value of 0 (low cost), 1 (medium cost), 2 (high cost) and 3 (very high cost)

Our data is preprocessed and pre-cleaned. There is no missing value in our dataset. Also, there is no outlier in our dataset.

CLASSIFICATION METHODS AND EVALUATION

- K-Nearest Neighbors
- Random Forest
- Support Vector Machine
- Neural Network

In our project, we use 60% data as the training set and the rest 40% as test set.

The criteria to compare each classification method is the **test accuracy**.

K-NEAREST NEIGHBORS

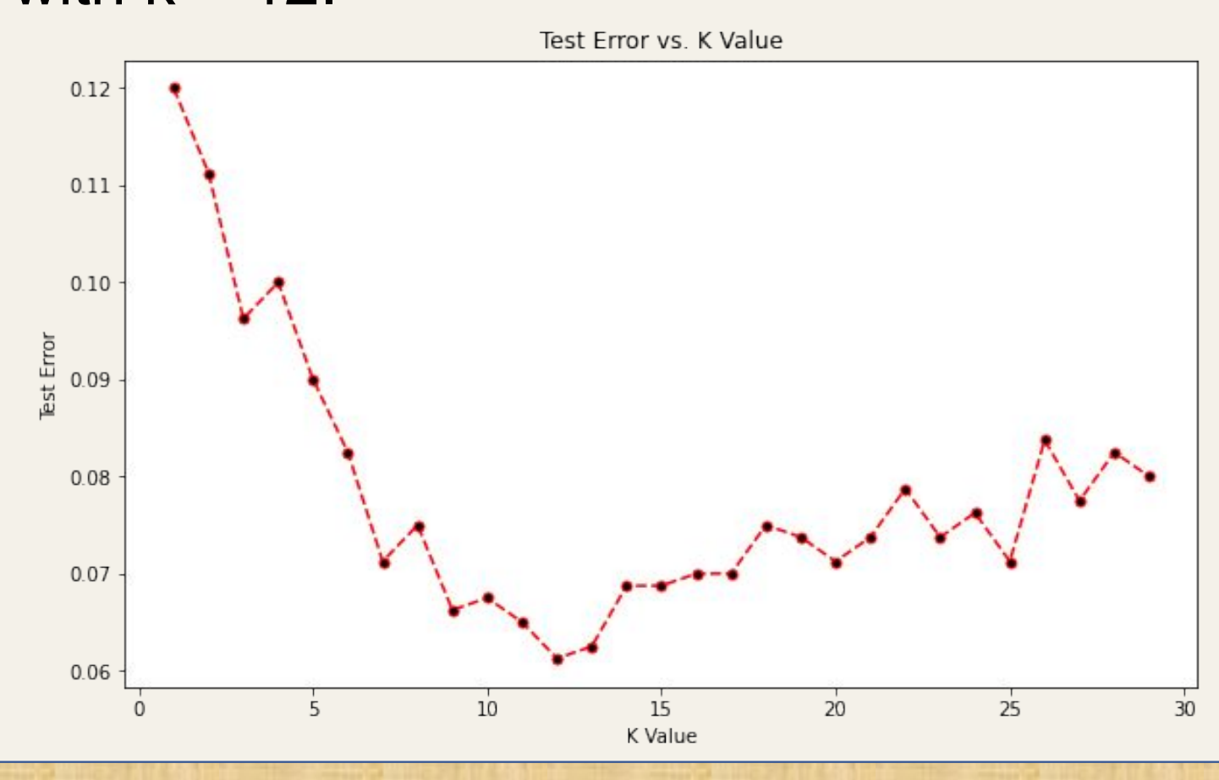
The k-nearest neighbors algorithm for classification in Sklearn Package measures the Euclidean distance between the target vectors and the train vectors. The classification result was determined by the majority vote of the nearest k neighbors. This is a **convex** program.

The Euclidean distance is calculated as:

$$distance(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

where x and y are data points in n-space.

The optimal k value was chosen by plotting different k values against their test error. In this project, the accuracy is **93.9%** with k = 12.



RANDOM FOREST

Random Forest is built on Decision Trees and predictions are the averaged prediction of the individual classifiers.

For Random Forest Classifier in Sklearn package, each tree in the ensemble is built from a bootstrapping sample from the training set.

We choose the two main parameters, number of estimators and number of max features, by test accuracy. Because the max accuracy is achieved when n_estimators = 100 and max_features = 14, we conclude that these two parameter optimized the model.

In this project, the accuracy is **92.1%**.

SUPPORT VECTOR MACHINE

The Support Vector Machine algorithm is embedded in the package "scikit-learn". In the dataset, there are 4 classes in "price range" but SVM is designed for binary classification problems. In this package, multi-class classification problem can be split into multiple binary classification problems and train a binary classification model each (OvO). Hence, this solves the **convex** primal program for each pair of classes and for 6 classifications (4 x 3/2), for class i and j:

$$\min \frac{1}{2}(w)^T w + C \sum_{i=1}^{n} (\zeta_i)$$

$$\text{subject to } \left((w)^T \phi(x_i) + b\right) \geq 1 - \zeta_i\,,$$

$$\left((w)^T \phi(x_i) + b\right) \leq -1 + \zeta_i\,, \zeta_i \geq 0$$

where $x_i \in R^p$, $\phi(x_i)$ maps x_i into a higher dimensional space and $y_i = \{-1,1\}^n$. And $C > 0$ is the regularization parameter. Here, let $C = 1$. The kernel function of radial SVM is:

$$K(x_i, x_j) = \exp\left(-\gamma ||x_i - xj||^2\right) = \phi(x_i)^T \phi(x_j)$$

where $\gamma = \frac{1}{20} = 0.05$ here since there are 20 features.

Then, solve the corresponding dual program to get the optimal outputs ω and b. And by this decision function:

$$\text{sgn}(\omega^T \phi(x_i) + b))$$

The predicted labels are gathered and recorded. The final predicted labels are determined by majority voting of predictions from 6 such dual programs.

So far, the major algorithm the package is using is **coordinate descent**.

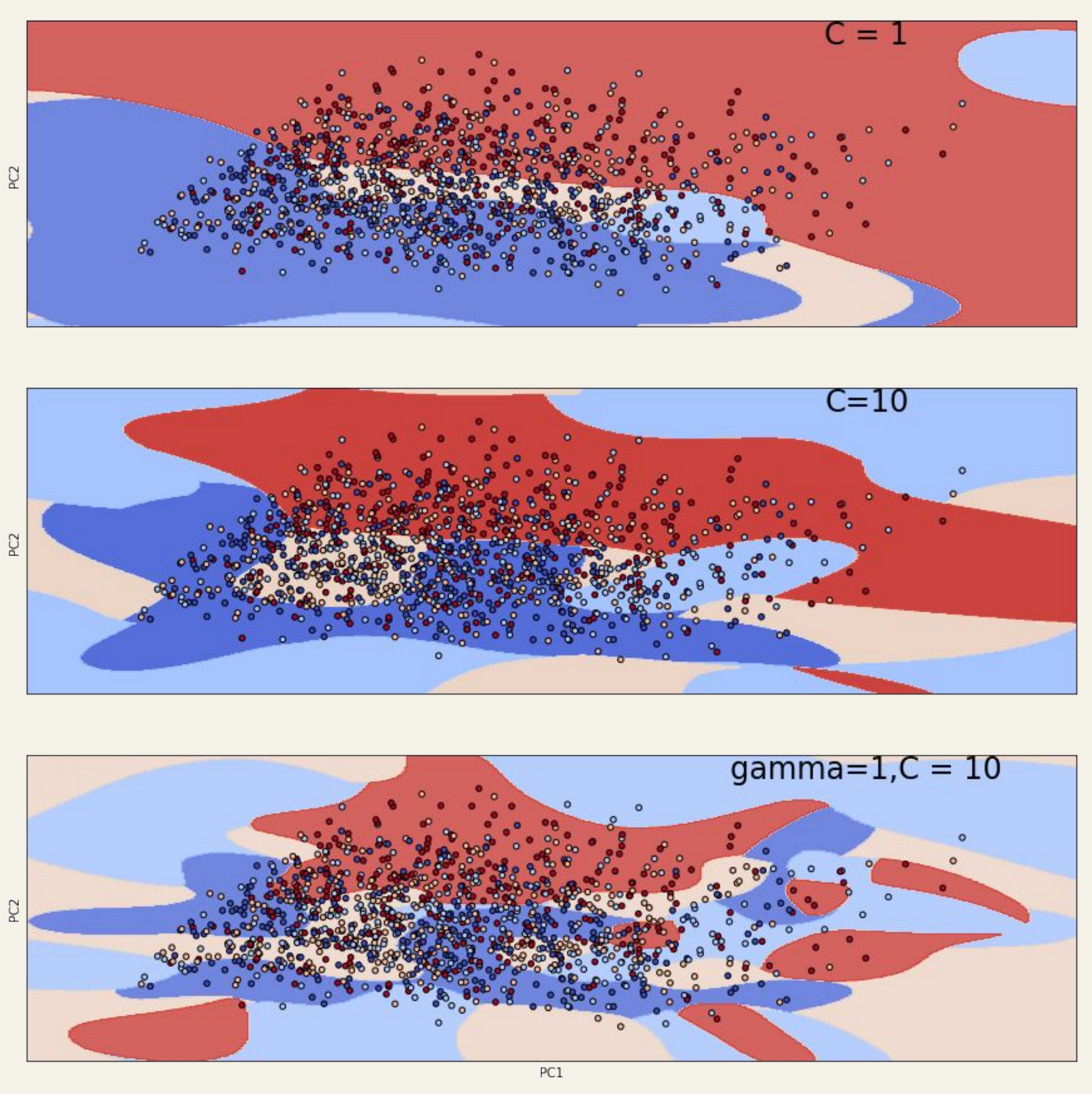
By python, the accuracy score of this method is: **94.8%.**

SVM IN SKLEARN

1. Linear SVM:
   Kernel SVM is a great method with higher accuracy than linear SVM but sometimes, for example, when the number of samples is small, there is no need to use kernel trick. Instead, the linear SVM will be more suitable. In sklearn package, there is also algorithm to solve this optimization program. With the same primal program as Kernel SVM, linear SVM only need to solve 4 such programs since it trains the dataset with OvR, not OvO.

2. Parameter C, gamma:
   C and gamma are two parameters of SVM in sklearn (only C if it is linear). Changing C and gamma will give a accurate classification result. The greater C is, the smoother and curlier the decision function will be. And the greater gamma is, the better to capture the shape of training vectors. Thus, increasing C and gamma leads to overfitting.

Compare different C and gamma



Neural Network

We build a sequential neural network to predict the price. We used three layers of neurons with dense 16, 12, and 4, respectively.

We use Relu as activation function for the first two layers and softmax as the output function. The Relu function preserves the non-negative part of the variable. The softmax function takes a vector of values and transforms it into a vector of values that sum up to 1.

We use Adam as the optimizer. Adam is a popular method that is used by stochastic gradient descent. Unlike regular gradient descent, stochastic gradient descent uses only a stochastic or random batch of training samples to update parameters. Thus, stochastic gradient descent is suitable when the dataset is very large.

The Adam optimizer does not only get stochastic gradient at each iteration, it also updates the first moment (the mean) and the second raw moment (the uncentered variance) accordingly.

The algorithm of the Adam optimizer is shown as below:

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

We also use dropout technique to avoid potential overfitting. Dropout randomly ignores some units and they will not be included in both forward and backward paths.

The accuracy of the neural network model on the testing set is **93.5%**.

CONCLUSION

Method	Accuracy
K-Nearest Neighbors	93.9%
Random Forest	92.1%
SVM	94.8%
Neural Network	93.5%

REFERENCES

Platt JC. Probabilistic outputs for SVMs and comparisons to regularized likelihood methods. http://research.microsoft.com/~jplatt. 1999 Mar 26. https://www.cs.colorado.edu/~mozer/Teaching/syllabi/6622/papers/Platt1999.pdf

Chang C-C, Lin C-J. LIBSVM: A Library for Support Vector Machines. https://www.csie.ntu.edu.tw. 2019 Nov 20.

Hsu C-W, Chang C-C, Lin C-J. A practical guide to support vector classification. Edu.tw. [accessed 2020 Dec 2]. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

sklearn.tree.DecisionTreeClassifier — scikit-learn 0.23.2 . [accessed 2020 Dec 2]. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Nearest Neighbors Classification | Statistical Classification . [accessed 2020 Dec 2]. https://www.scribd.com/document/421227117/Nearest-Neighbors-Classification

Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv.org. 2017 Jan 30 [accessed 2020 Dec 5]. https://arxiv.org/abs/1412.6980

Contact Information

Ella Wang (yw701@georgetown.edu)
Jingjie Ma (jm3292@georgetown.edu)
Han Zhang (hz274@georgetown.edu)