Data Science - Homework 6 (Regression)

- Introduction of datasets

1. Adult Dataset (from UCI)

An individual's annual income results from various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, and etc. This dataset also known as "Census Income" dataset. Therefore, the goal of this assignment is to accurately predict whether an adult makes more than 50K in an year on the basis of the features given.

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss	hours_per_week
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40

First, after importing the dataset, I found that it has 32561 records in training set, and test set has 16282 records, both of them has 15 attributes. Next, I check the details of training set, as can be seen from the figure on the right that there are missing values in three columns, including workclass, occupation and native_country. I will drop the rows with missing values.

```
df_adult_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age               32561 non-null int64
workclass         30725 non-null object
fnlwgt            32561 non-null int64
education         32561 non-null object
education_num     32561 non-null int64
marital_status    32561 non-null object
occupation        30718 non-null object
relationship      32561 non-null object
race              32561 non-null object
gender            32561 non-null object
capital_gain      32561 non-null int64
capital_loss      32561 non-null int64
hours_per_week    32561 non-null int64
native_country    31978 non-null object
income_bracket    32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

2. Mobile Price Dataset (from Kaggle)

If we want to start our own mobile companies, an important thing we must know is to how to estimate price of mobiles our companies create. In this competitive mobile phone market you cannot simply assume things. The aim of this dataset is to solve this problem, it contained sales data of mobile phones of various companies. This dataset is a very small dataset, the training set and test set contain 2000 records and 1000 records respectively, and neither has any missing values.

- Use Python to perform regression analyze the some datasets and explain the results you obtain.

1. Adult Dataset (from UCI)

In the process of data cleaning, I ran a for loop over all the columns which gets the count of unique values. I observed that fnlgwt attribute has a lot of distinct values (around 20000+ values), it may be noisy data for my model. And I found that the two columns education and education_num have the same meaning. In summary, I removed the fnlgwt and education.

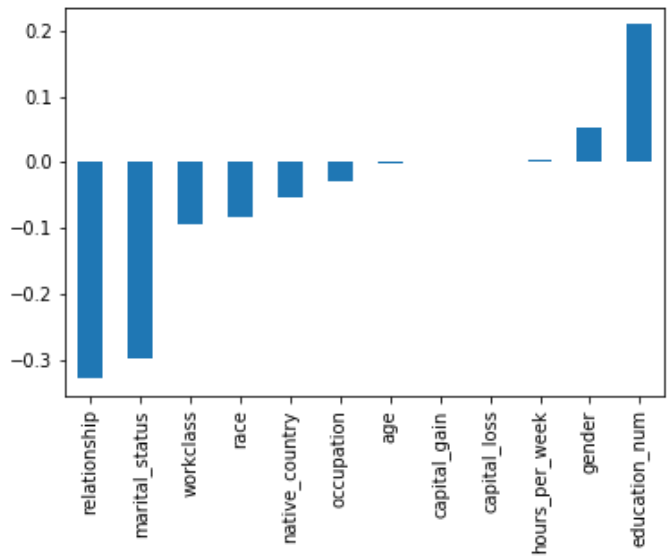
This dataset is a mixture of numerical and categorical data types, where the non-numerical columns are represented using strings, so I need to convert categorical data to something that the machine could better understand, the LabelEncoder that sklearn provided is a helpful method.

Move on to regression analysis, compared to linear regression, I think that logistic regression is a binary classifier algorithm, I will use it as my first regression model.

The accuracy score is around 0.80, we might get a slightly different number each time because it is a stochastic process. But my interest here is the coefficient of the features in the logistic regression function, to rank the importance of predicators.

```
lr = LogisticRegression()

lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
score = accuracy_score(y_test, y_pred)
print(score)
```

```
0.799867197875166
```




```
education_num     0.209168
gender            0.052393
hours_per_week    0.004648
capital_loss      0.000693
capital_gain      0.000319
age              -0.000777
occupation       -0.028468
native_country   -0.052371
race             -0.082381
workclass        -0.094194
marital_status   -0.298435
relationship     -0.328509
dtype: float64
```

As can be seen from the graphs and data on the previous page, this ranking is somewhat useful. It tells us features such as education, gender, work hours, capital-gain/loss are more important than others like occupation, age and native country when it comes to predicting the income of a person. And the relatively low ranking of occupation seems counterintuitive. Moreover, what does it mean that the relationship and marital-status negatively impact the income level?

Due to the above doubts, I wanted to implement the other model, the GBRT, it is a integrated learning architecture of the boosting, which uses the forward distribution algorithm to select an appropriate decision tree function based on the current model and fitting function, thereby minimizing the loss function. Unlike bagging, the tree generated by the boosting architecture will be related to the previous tree. How to illustrate whether the prediction model has better accuracy in describing the experimental data? I chose the MSE for this dataset.

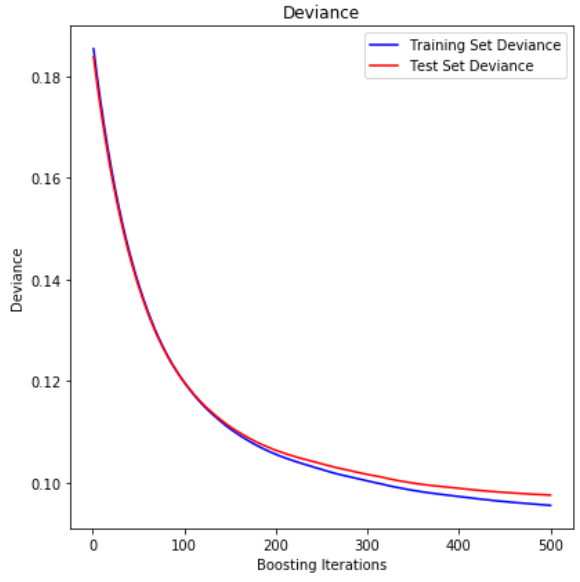
The smaller the value of MSE, the better the accuracy of the prediction model in describing the experimental data. Therefore, I can see that MSE value of GBRT is very small, its predictions have few outliers, which means that it has a high accuracy for predicting this issue.

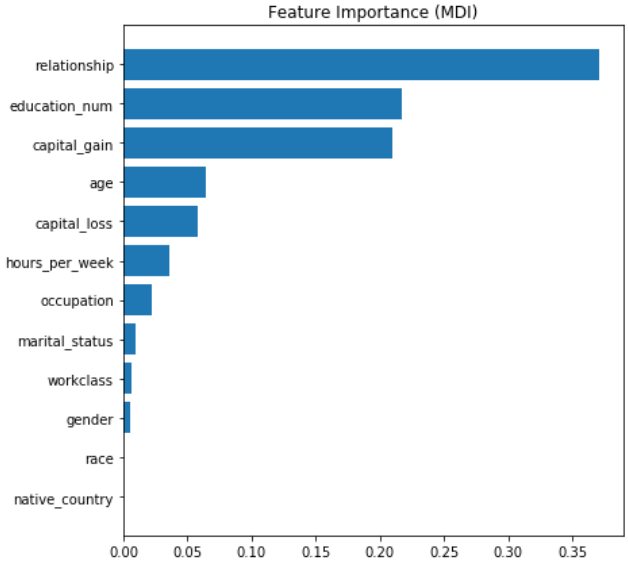
```
params = {
    "n_estimators": 500,
    "max_depth": 4,
    "min_samples_split": 5,
    "learning_rate": 0.01
}

reg = ensemble.GradientBoostingRegressor(**params)
reg.fit(X_train, y_train)

mse = mean_squared_error(y_test, reg.predict(X_test))
print("The mean squared error (MSE) on test set: {:.4f}".format(mse))
```

```
The mean squared error (MSE) on test set: 0.0975
```

```
rmse = mean_squared_error(y_test, reg.predict(X_test), squared = False)
print("The root mean squared error (RMSE) on test set: {:.4f}".format(rmse))
```

```
The root mean squared error (RMSE) on test set: 0.3123
```





From the feature importance ranking of GBRT, there is a big difference with the results of the logistic regression, especially in the relationship ranking. Feature importance provides a highly compressed, global insight into the model's behavior, but in some cases it has some problems. The feature importance depends on shuffling the feature, which adds randomness to the measurement. When the permutation is repeated, the results might vary greatly.

```
# 10-fold cross-validation with three features
feature_cols = ['relationship', 'education_num', 'capital_gain']
X_test_select = df_adult_test[feature_cols]
print (-cross_val_score(reg, X_test_select, y_test, cv=10, scoring='neg_mean_squared_error').mean())
```

```
0.10874528117999475
```

Because the MSE value calculated by only using the three features of relationship, education, and capital_gain is large, it can be known that the prediction result obtained by using all the features is better.

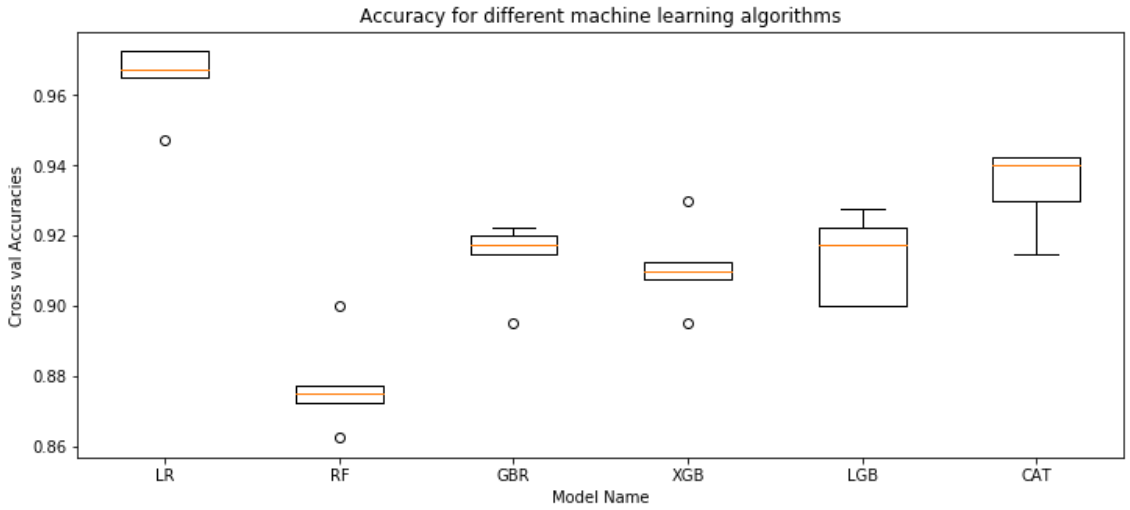
2. Mobile Price Dataset (from Kaggle)

Compared to the first dataset, I decided to use more regression models to analyze for the second dataset, and then the evaluation indicators is different from the previous MSE. Here I want to emphasize and compare the interpretability of each model, so I use the R2 (R-squared) to evaluate.

On the other hand, to avoid the problem of permutation feature importance, a alternative to permutation feature importance are variance-based measures. SHAP importance has similarities to a variance-based importance measure, if changing a feature greatly changes the output, then it is important. Permutation feature importance is based on the decrease in model performance. SHAP is based on magnitude of feature attributions.

```
results = []
names = []
for name, model in models:
    pipe = Pipeline([('ct', ct), (name, model)])
    scores = cross_val_score(pipe, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1, verbose=0)
    names.append(name)
    results.append(scores)
    print("model %s accuracy: %.4f variance: %.4f"%(name, np.mean(scores), np.std(scores)))
```

```
model LR accuracy: 0.9650 variance: 0.0092
model RF accuracy: 0.8775 variance: 0.0123
model GBR accuracy: 0.9140 variance: 0.0098
model XGB accuracy: 0.9110 variance: 0.0112
model LGB accuracy: 0.9135 variance: 0.0115
model CAT accuracy: 0.9340 variance: 0.0106
```



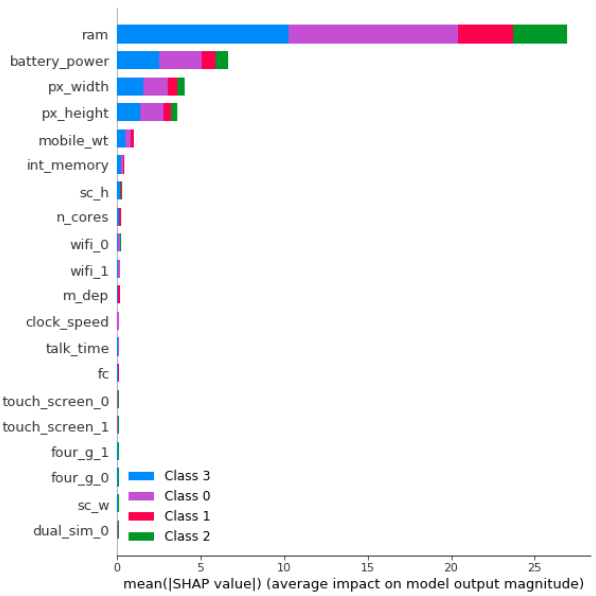
I use K-fold cross validation (cv=5) to validate these models. To apprehend at a glance, I think the boxplot that taught in the previous courses is a good way to present the accuracy of these models. We can observe that logistic regression has the best accuracy, and random forest has the worst accuracy.

Unlike MSE, R2 evaluates the variance, which is used to indicate the degree of variation of the model error. The value for R2 can range from 0 to 1. 0 indicates that the response variable cannot be explained by the predictor variable at all. 1 indicates that the response variable can be perfectly explained without error by the predictor variables.

```
model LR R2: 0.9683
model RF R2: 0.8941
model GBR R2: 0.9226
model XGB R2: 0.9302
model LGB R2: 0.9282
model CAT R2: 0.9506
```

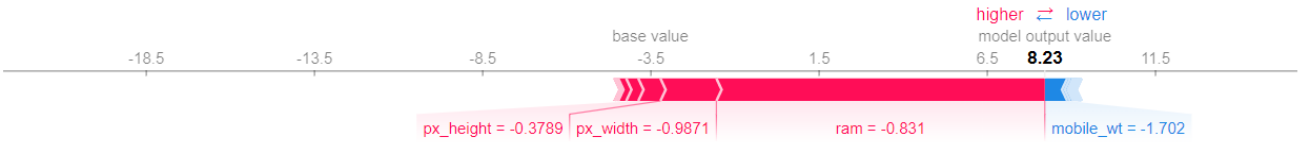
From the above, only random forest has R2 value below 0.9. I can say the predictor variables of these models are able to predict the value of the response variable precisely.

Finally, in the feature importance part, I obtained the model interpretation using SHAP. Then, I chose to use logistic regression to implement this method. We can clearly see that only four variables are very important and influencing the prediction, while rest of the variables have no importance.



The following figure shows SHAP explanation force plots in this dataset. The features that were important to making the prediction for this observation are shown in red and blue, with red representing features that pushed the model score higher, and blue representing features that pushed the score lower. Features that had more of an impact on the score are located closer to the dividing boundary between red and blue, and the size of that impact is represented by the size of the bar.

Take class=0 as an example,



We can straight away see that SHAP value model interpretability is very effective. It explains the variable contribution in additive sense which is easier to grasp and also shows which variables are influencing the decision.

- Discuss possible problems you plan to investigate for future studies.

After analyzing this two datasets, a problem that I was very concerned about is the approach of encoding categorical features to ordinal variables is intrinsically flawed. It works fine when the feature has a binary nature, e.g. gender. However, when the feature has more than two categories, the model maybe will make some arbitrary connections between unrelated values.

For example, in the workclass of the adult dataset, if I assign “1” to “Private”, “2” to “Sel-emp-not-inc”, and “3” to “Local-gov”, the model maybe will think “Private” are more similar to “Sel-emp-not-inc” than “Local-gov” because “1” is closer to “2” compared to “3”. But this is not my expectation.

I think I can change to use one-hot-encoding to solve this problem, converting feature A to vector [0,0,1], feature B to [0,1,0], and feature C to [1,0,0].