THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Exploration of Machine Learning Algorithms Through house price dataset (DATA7703)

Course Coordinator:

A/Prof. Marcus Gallagher

Lecturer:

A/Prof. Marcus Gallagher

Dr. Nan Ye

Students:

Ashama Plipat (46925491)

Shuo Yuan (46920348)

Jia Ji (47206085)

Nikolaos Chatzis (47559932)

Ritika Rana (47306725)

BRISBANE CITY, NOVEMBER 2022

We DO NOT give consent for this to be used as a teaching resource.

Abstract

House price has been rising every year all over the world. The aim of this project is to use machine learning methods that had been taught in DATA7703/2022 to optimize appropriate performance, predicted house price. In this report, house price dataset from Kaggle competition would be used with different models, as following, PCA, t-SNE, Polynomial regression, Random Forest regression, Support Vector Machine (SVM) regression, Naïve Bayes, k-Means, Gaussian Mixture, k-NN and Random Forest classification. Finally, limitations and further discussion will be discussed according to the models and the dataset.

Table of Contents

Introduction

With the rapid development of social economic and people's living standard, the degree of urbanization has been continuously improved and housing market investment has become one of the most popular industries nowadays. House price prediction can help the developers determine the selling price of different houses and also help people schedule the right time for buying a house so that they could plan their finances well according to the future price range. In addition, house price prediction is also beneficial for real estate investors to know the price trend of a certain place or for house buyers who need to waste a long time in the inspections with housing agent to accumulate experience, which is quite a time-consuming process, we can directly use quantitative data to help them establish a basic, clear concept.

The significance of house price prediction and the trend of constant changes of house prices really aroused our group members' interest. After searching from Kaggle, we found a dataset about house prediction which has 79 explanatory variables of residential homes in Ames, Iowa. Ref: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview/description.So, our group members decided to analyze this dataset using the technical methods we learned during the course of machine learning to fit and optimize all the possible models and compare the outputs to figure out the most reasonable one which could be applied in this real-world problem.

During our project, since we already had a complete dataset in csv format from Kaggle, we applied Polynomial Regression, Random Forest Regression, EDA techniques, Gaussian mixture models, PCA, t-SNE and SVM regression to attempt to find the best one after comparing different models' results. From the consequences, we aim to figure out the best regression model applied in our dataset for house price prediction. And this kind of model might also work well on other house price prediction datasets, while if the model fits our dataset well but not well enough on others, this might be an indicator of underlying differences in economy/culture of the two places. Through our results, we really hope that we could provide useful information for other researchers, relevant departments and even investors who are interested in this field in the future. Besides that, even though our dataset might not be suitable for classification in this real-world problem solving, we

still applied some classification methods we learned before on this dataset which could be a review part of this course in our project.

In the next parts, we would have a detailed introduction about the specific procedures through our whole project, the techniques we applied to deal with this dataset and the consequences we obtained from our project.

Methodology

Data-preprocessing

Concept

Imputation for missing value:

When dataset have missing values, if the missing data are not handled properly sometimes it could cause problems for analyzing data or fit model and it could make the model biases. So, imputation is seen as a way to avoid this incident. Besides, it requires careful planning and attention. Imputation preserves all cases by replacing missing value with an estimated value based on other available information. When all missing values are handled by imputation and replaced, the dataset can be used to analyze or fit model by machine learning techniques.

There are many techniques embraced by scientists to account for missing data. In this report, simple imputation, linear regression imputation and nearest neighbor imputation will be discussed.

Mean imputation is quite straightforward. It is a method that missing values are replaced by the mean of available cases. The advantage of this method is easy to use, and the sample size remains the same. However, the variability in the data is reduced, so the standard deviations and the variance estimates tend to be underestimated. This method usually causes biased estimates.

Regression imputation, this method replaces missing values by a prediction from a regression equation. Regression imputation uses all complete observations information in the dataset to

predict the values of the missing observations. Imputed values in regression imputation fall on regression line. This method will overestimate the correlations. However, the variances and covariances are underestimated. Normally, regression imputation is classified into two different versions which are deterministic and stochastic regression imputation. In this report, we will discuss about deterministic regression. Deterministic regression imputation replaces missing values with the exact prediction from a regression model without considering a random variation around the regression slope. It could make imputed values too precise and lead to overestimation of the correlation between x and y.

k-NN imputation will identify the neighboring points through a measure of distance and missing values can be estimated by using completed values of neighboring observations.

In this report, our dataset is from Kaggle (https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques), there are 79 features, or 79 columns, missing data spread to 19 columns. However, there are some columns that look unusual. For example, in Pool quality column there are 1453 missing values while there are 1460 row of data, which are not usual. After investigation, it turned out NA value for some missing data is not a missing value. So, for some columns NA is not available such as NA in Pool quality means no pool. Then, NA for some columns is replaced by not available for that specification. For example, NA in Pool quality is no pool and NA in fence is no fence. Finally, there are only 5 columns that have missing data.

Table 1: amounts of missing value for different feature

Columns name	**Amount of missing value**
LotFrontage: Linear feet of street connected to property	259
MasVnrType: Masonry veneer type	8
MasVnrArea: Masonry veneer area in square feet	8
Electrical: Electrical system	1
GarageYrBlt: Year garage was built	81

Then, 3 imputations techniques applied to replace missing values. The result of each technique can show below. I used cross-validation techniques to identify which imputation has the best

performance. With parameter of k-fold equal to 10 and use Random Forest regressor to fit a prediction model. Then, choose the best average cross-validation value.

1. Simple imputation

For missing data, there are two type of simple imputation which are mean imputation and most frequent imputation. Mean imputation replaces missing values by average value of all observations while most frequent imputation replaces missing values by most frequent value.

The figure below is a relation between missing data and SalePrice, which is output. The blue dots are original data, and the green dots are imputed values by mean imputation.

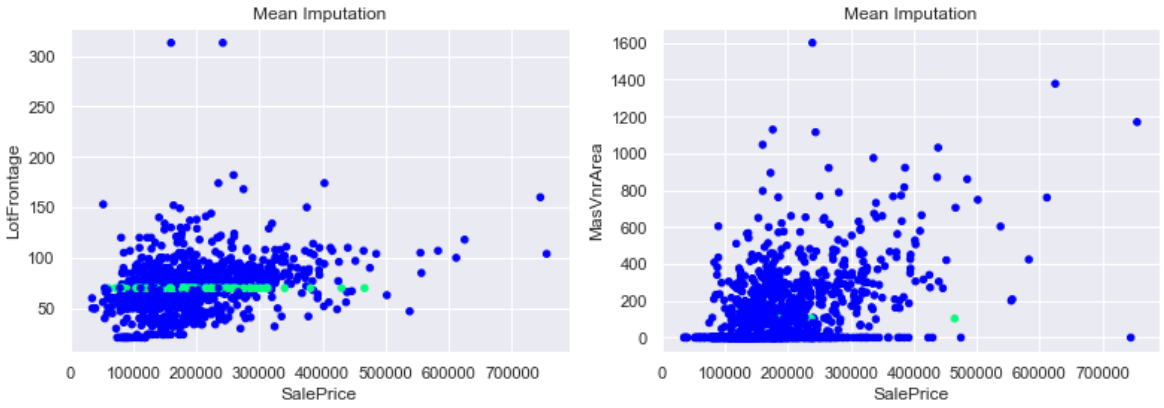
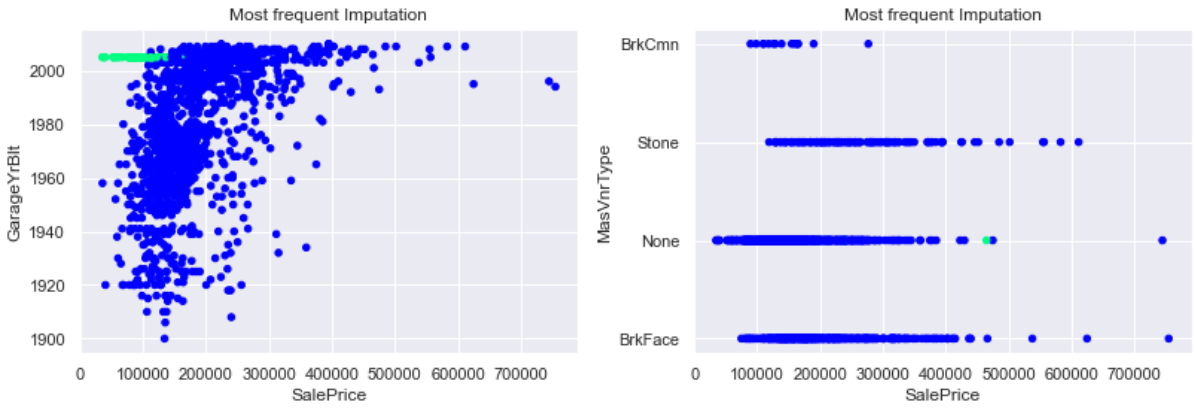


Figure 1: an example of x and y relation of mean imputation

The figure below is a relation of data and SalePrice and impute by most frequent imputation. Blue dots are orginal data and the green dots are imputed valued by most frequent imputation.



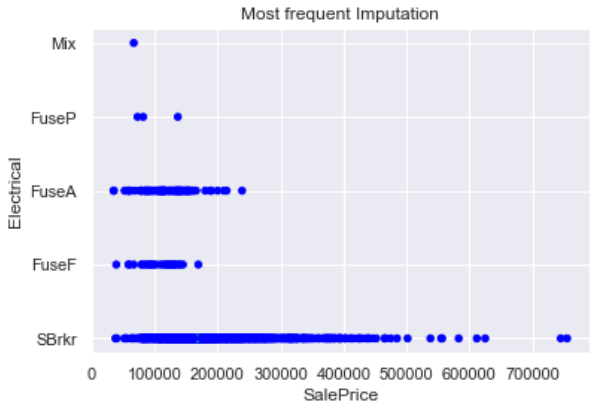
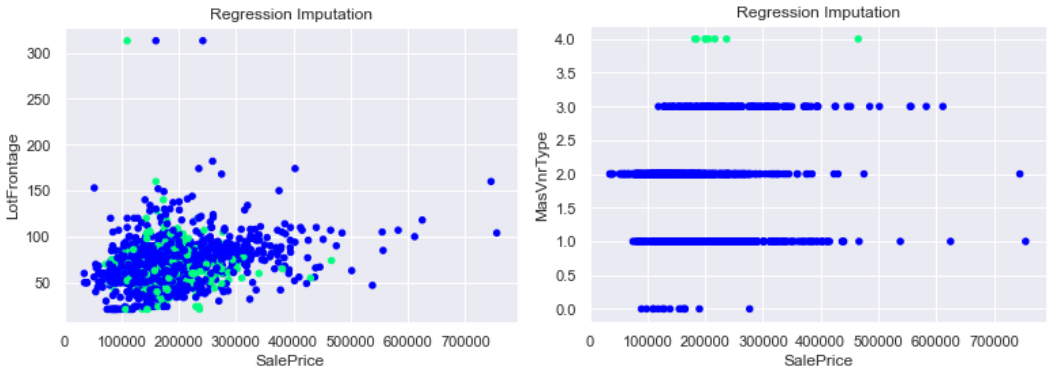


Figure 2: an example of x and y relation of most frequent imputation

This technique, we combine two methods together which are mean imputation and most frequent imputation. Using mean imputation for numerical value while most frequent imputation for categorical value.

2. Regression Imputation

For this method, we use deterministic regression imputation to replace missing data by predict missing value. The figure below shows relation between missing columns and SalePrice, blue dots are data without null and green dots are imputed data by deterministic regression imputation.



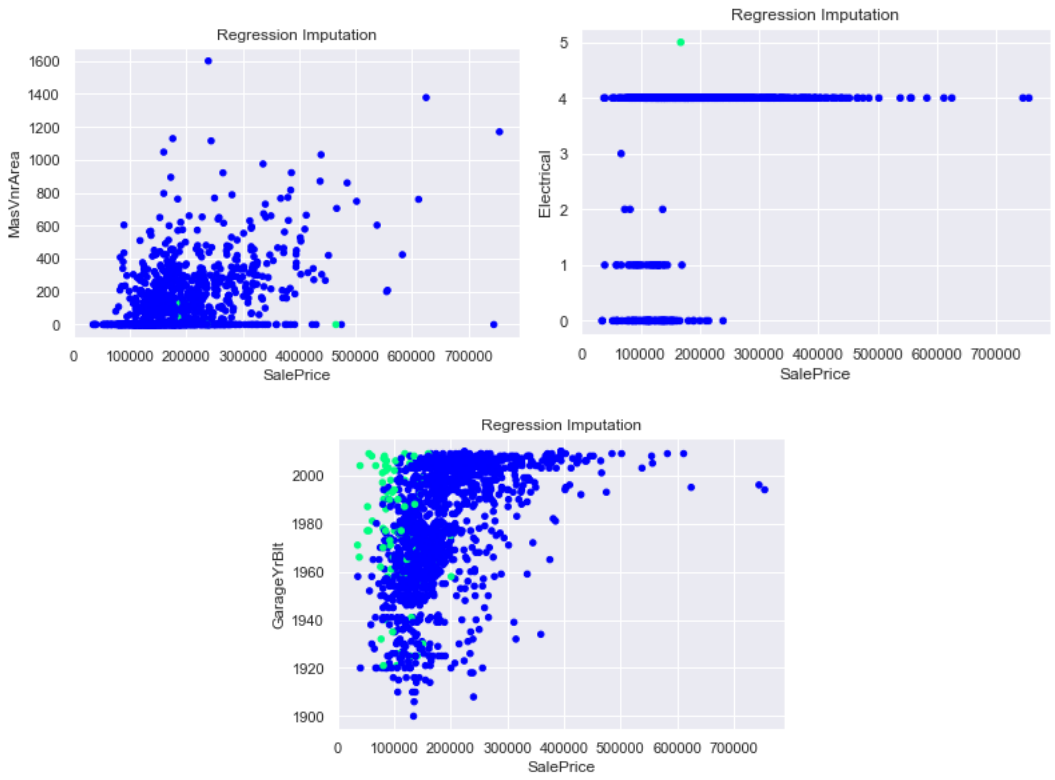
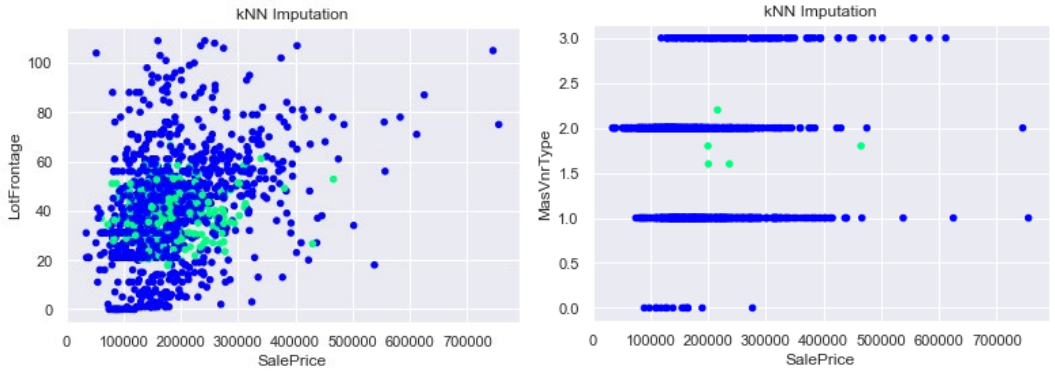


Figure 3: an example of x and y relation of regression imputation

3. kNN imputation

While kNN algorithm identify missing value by neighboring observations. The figure below demonstrates a relation between data of missing columns and SalePrice by kNN imputation.



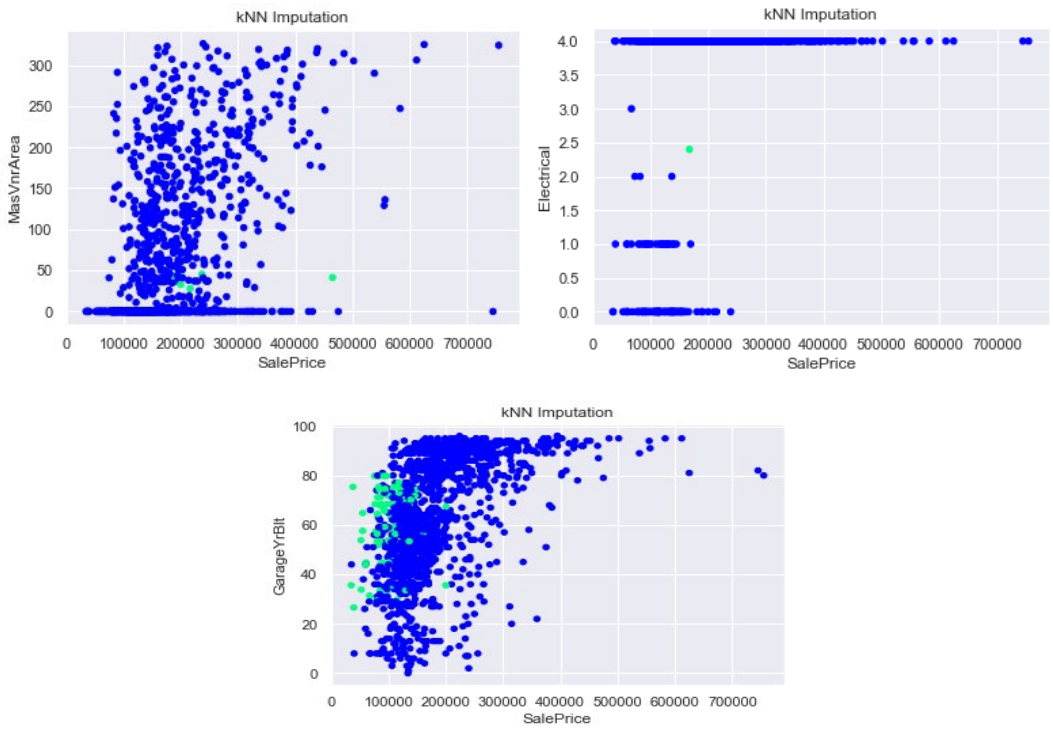


Figure 4: an example of x and y relation of kNN imputation

Finally, before we choose which imputation to use. I use cross-validation to check which imputation has the best performance. We use ***sklearn.model_selection*** with ***KFold***, ***cross_val_score*** to define k-fold value and get cross-validation score after cross-validation. Then, we use ***sklearn.ensemble*** with ***RandomForestRegressor*** to fit the model for helping determine average of cross-validation score to find the best performance.

Before fit the model, we have to encoding all the data for the model to be able to fit it. There are two encoding methods that we use. The first one is label encoding and another one is one-hot encoding. Then, we use cross-validation with algorithm to find which combination has the best performance and we will use that combination, imputation, and encoding, to create models.

The table below is average cross-validation of different imputation and encoding.

Table 2: average cross-validation of different imputation and encoding

Imputation technique	**Average Cross-validation**
Simple Imputation with label encoding	0.8643042006489642
Regression Imputation with label encoding	0.99667438990707
k-NN Imputation with label encoding	0.8627718985715204
Simple Imputation with one hot encoding	0.9961043568738924
Regression Imputation with one hot encoding	0.8536816658802934
k-NN Imputation with one hot encoding	0.8545871349135302

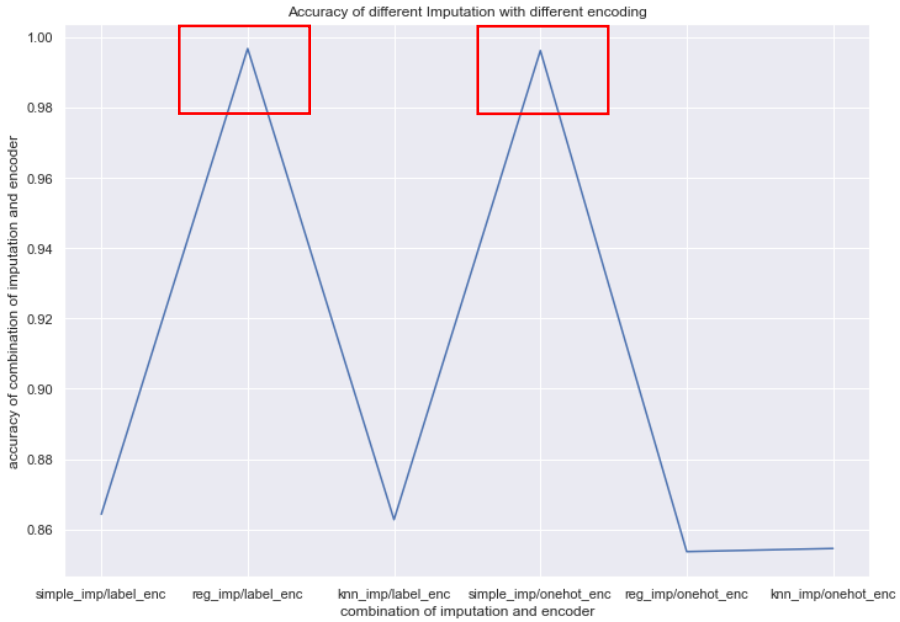


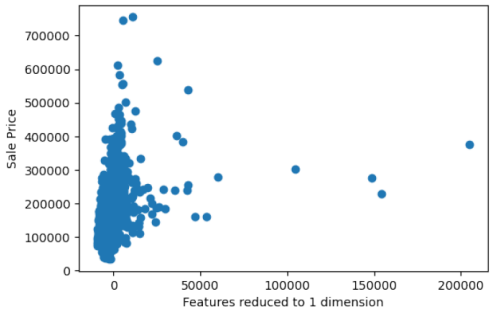
Figure 5: accuracy of different imputation and encoding

In the graph and the table above, we can see that there are two combinations that have similar score, regression imputation with label encoding has 0.99667438990707 score while simple imputation with one hot encoding has 0.9961043568738924. According to it, we can assume that both methods can be used because the performance is very similar. However, one hot encoding has a lot of columns, it could lead to a slow execution and cost a lot of time. For example, simple imputation with one hot encoding has 894 columns whilst simple imputation with label encoding only has 79 columns. Hence, we decide to use regression imputation with label encoding.

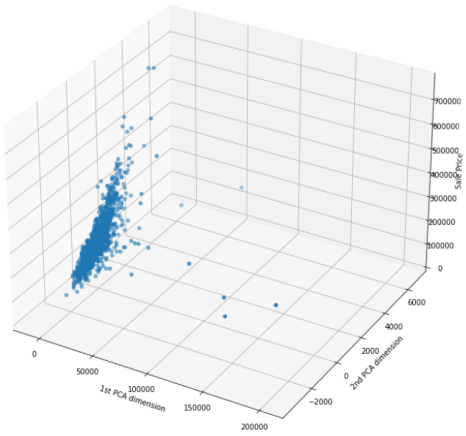
Dimensionality reduction PCA

Introduction

We begin our exploration of dimensionality reduction with PCA. For the purposes of visualization, we remove sale price from the dataset and we perform PCA on the remaining features. We provide plots for the first and second PCA components:



(a) First PCA component with sale price



(b) Two PCA components with sale price

Variance explained

We provide the scree graph of the data set. The y-axis represents the percentage of variance explained and the x-axis is the PCA components. We observe that by the third component we have already explained more than 99% of the variance and by the fourth component more than 99.9% of the variance.

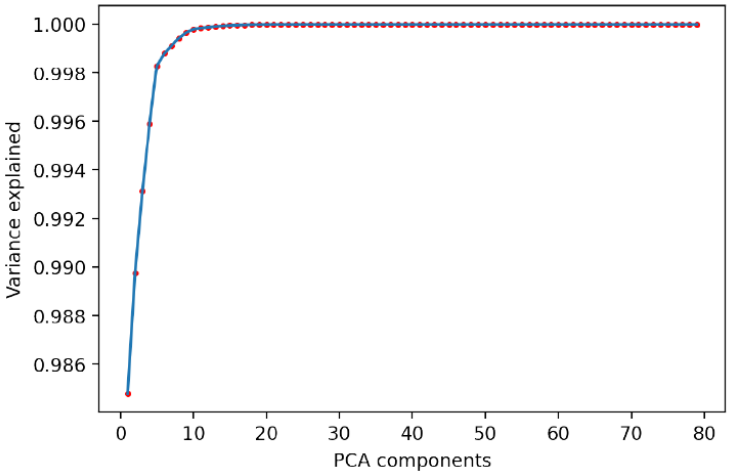


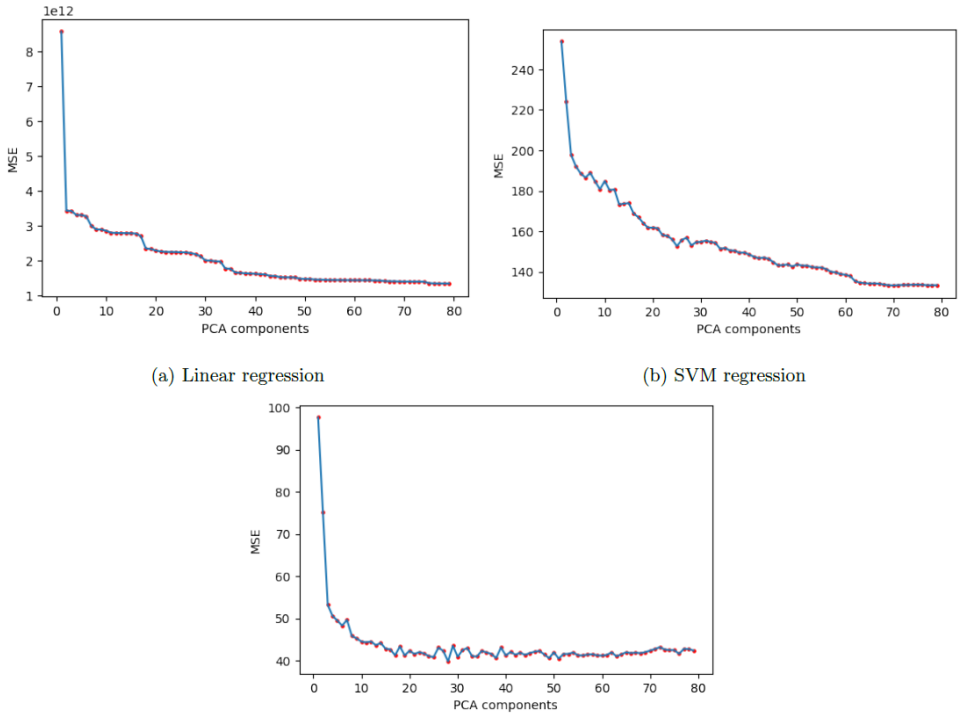
Figure 6: Screen plot

Performance of regression algorithm

In the previous section we saw that 4 PCA components already explain more than 99.9% of the variance. However, does that mean that regression algorithms only need a few PCA components as well?

To test this claim we set up the following experiment:

- We will test Linear, SVM and Random Forest regression
- SVM regression has rbf kernel, C = 1, $\epsilon = 0.1$ (sklearn default options)
- Random Forest regression has max depth = 10
- For each PCA component we run the algorithms with Linear (raw data), SVM and Random Forest (standardized data) and record the MSE



(a) Linear regression

(b) SVM regression

(c) Random Forest regression

Conclusions

We begin with a disclaimer: the choice of hyperparameters in this section was not our object of study and we explore more on this on later sections. Hence, not a lot of emphasis was given on their choice, they just remained fixed throughout the experiment.

We observe that all algorithms follow the trend that more PCA components results to better performance. The rate that this happens changes from algorithm to algorithm.

T-SNE

Introduction

In t-SNE, the lower dimensional output of the data aims to preserve similarities. Similarities in the higher dimension space are measured using the SNE algorithm (proceeding t-SNE) and in the lower dimensional output, similarities are measured using a t-distribution with one degree of freedom.

There is one hyperparameter in the algorithm, perplexity. We present the output of t-SNE for varying values of perplexity.

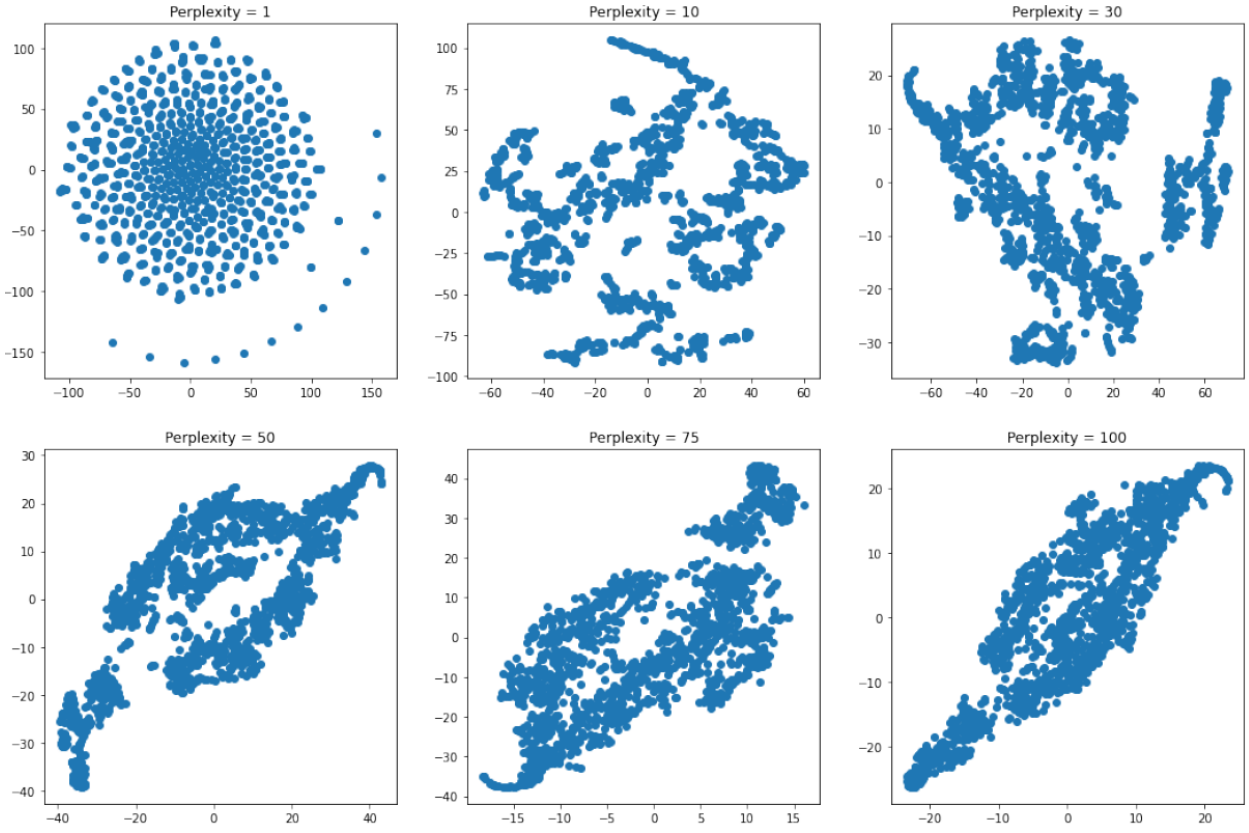


Figure 7: T-SNE

Formulating experiments

We are now ready to formulate two experiments, based on what we see in the output of t-SNE.

Experiment 1: Is there a cluster?

We immediately notice that the t-SNE output appears to have something that looks like a cluster and it remains consistent regardless of the perplexity value. This is an indicator that a cluster might exist in the original (high dimensional) data set.

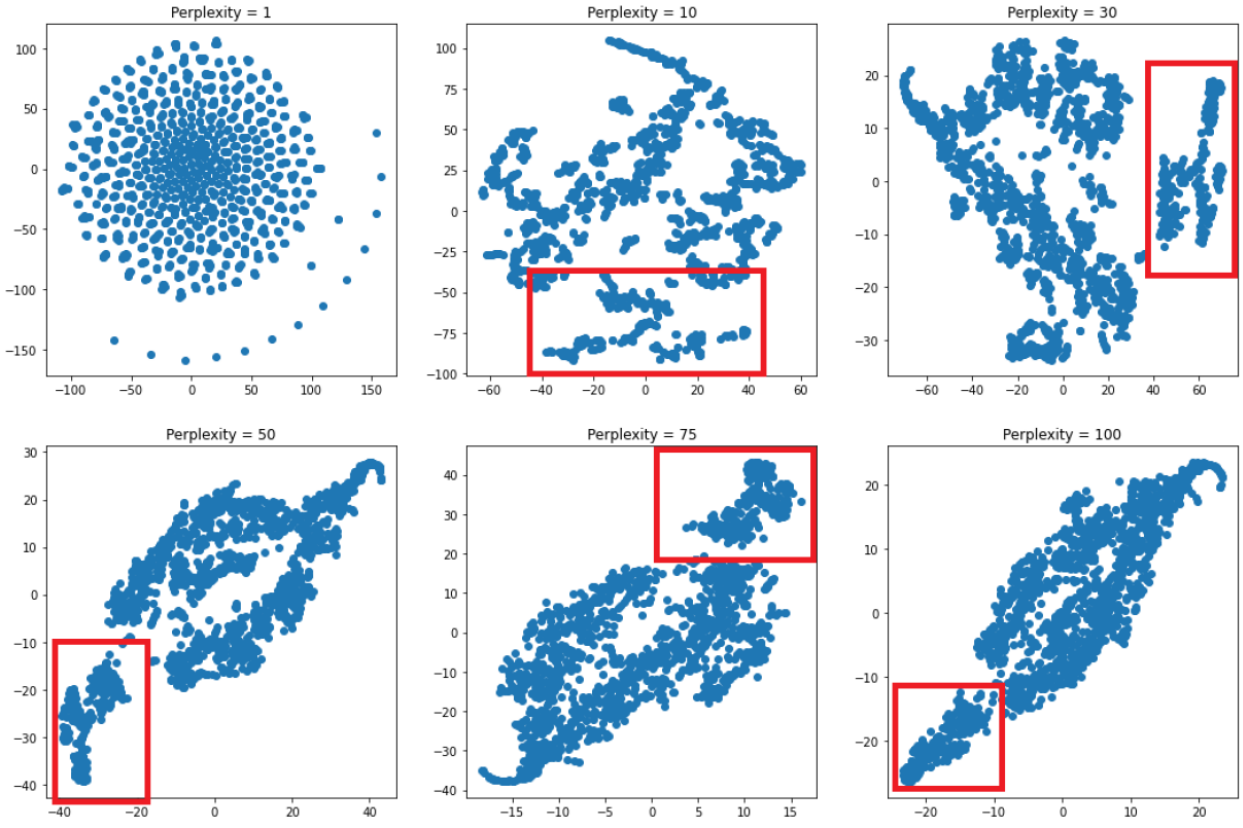


Figure 8: Cluster

Experiment 2: Does the cluster relate to Sale Price?

Given that we have observed the cluster in the t-SNE output we will formulate an experiment to see if it relates to sale price. This will be done by introducing a filtration.

t-SNE experiments

Experiment 1: a cluster in high dimensions?

For the purposes of this section we will fix perplexity to 30. We will also assign an artificial clustering based on what we perceive to be a cluster on the t-SNE output. Then we shall proceed as follows:

- Save the clustering labels in the lower dimensional space
- Perform Gaussian mixtures and kMeans (both with 2 components) on the higher dimensional data and save the output labels
- We define accuracy as the overlapping percentage of our artificial labeling and the algorithm outputs We repeat steps two and three 100 times with random starting points, we plot the results

Artificial clustering

We begin by constructing the artificial clustering

```
def mark_point(x):
    if x[0]>37:
        return True
    else:
        False

marked =[1 if mark_point(x) else 0 for x in data_transformed]
```

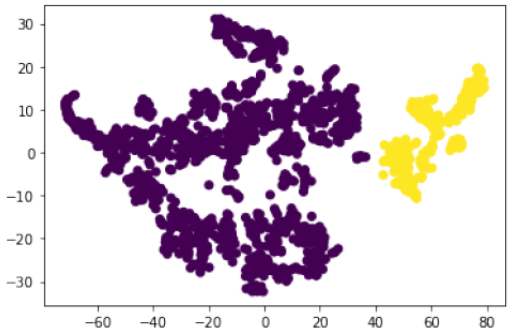


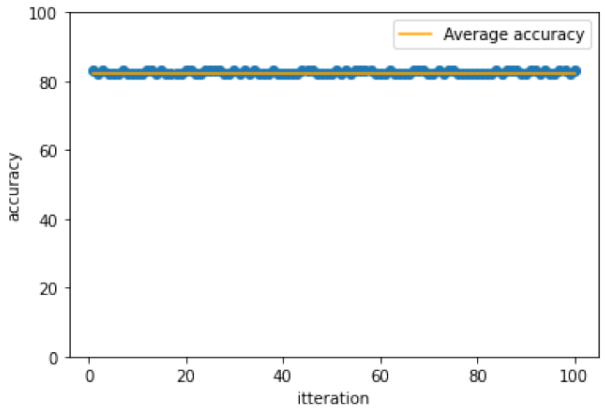
Figure 9: artificial clustering

Running the experiment

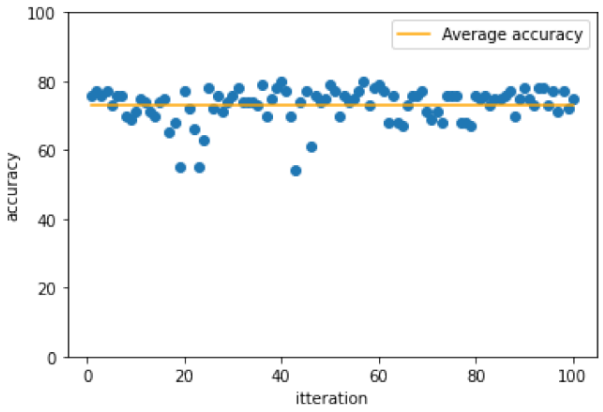
We now proceed with the experiment

```
#kMeans
results = []
for i in range(100):
    model = KMeans(init ='random',n_clusters = 2).fit(data)
    labels = model.labels_
    acc = [1 if labels[i] == marked[i] else 0 for i in range(len(marked))]
    acc = int(sum(acc)/len(acc)*100)
    results.append(max(acc,100-acc))


#Gaussian mixtures
results = []
for i in range(100):
    model = GaussianMixture(init_params ='random',n_components = 2).fit(data)
    labels = model.predict(data)
    acc = [1 if labels[i] == marked[i] else 0 for i in range(len(marked))]
    acc = int(sum(acc)/len(acc)*100)
    results.append(max(acc,100-acc))
```



(a) kMeans



(b) Gaussian mixtures

Conclusions

We observe that the algorithms we tested on the high dimensional data set, have very high accuracy with respect to our artificial clustering (between 75% and 85%, depending on the algorithm). From this we conclude that what we observed as a cluster in the t-SNE output, indeed exists in the original data set.

Experiment 2: Does the cluster relate to price?

In this section we will test whether the cluster relates to price, for example the is the cluster part of the lowest or highest sold houses, or does it exist independently.

Formulating the experiment

We again fix perplexity to 30. We claim we can answer our question by a filtration of the data set w.r.t. sale price. At each step we plot with bright red an increasing percentage of the houses with the lowest sale price and the rest with blue. For example in one step we plot the lowest 10% and in the next the lowest 20%. We proceed like this all the way to 100%.

To prove our claim that this is sufficient to check if the cluster relates to sale price, we refer to the following thought experiment. Assume that the cluster represents the houses of price between 50-60% (ie. some average range houses). At filtration level 50% all the cluster is blue. At filtration level 60% all the cluster becomes red. This sudden jump in color distribution indicates proves the relation of the cluster with the sale price. Now on the contrary assume that the cluster does not relate to sale price. Then we expect no sudden jumps in color distribution as in the previous case. We expect to see the cluster turn from blue to red gradually.

We are now ready to plot our results:

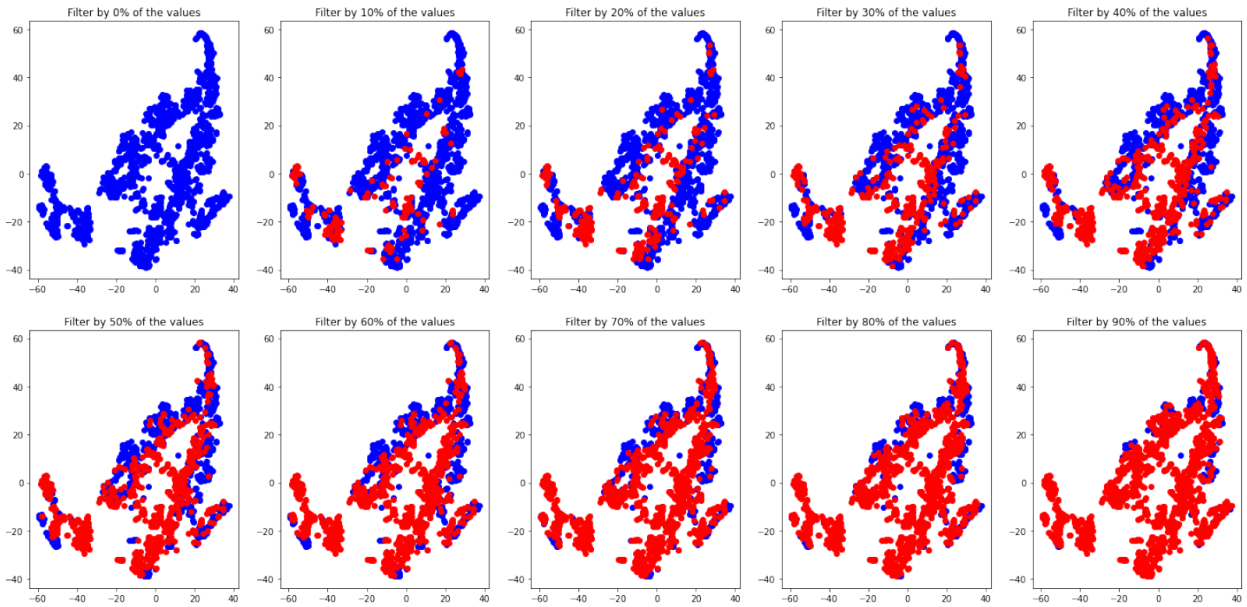


Figure 10: Filtration

Conclusions

We observe the second scenario we discussed. We see that the cluster (bottom left corner) gradually turns from blue to red, with no sudden shifts in color distribution. This proves that the cluster does not relate to sale price.

Predictive model

1. Regression model

1.1. Polynomial regression

Concept

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x. With an equation:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon$$

Conveniently, this model is linear from the point of view of estimation, since the regression function is linear in terms of the unknown parameters β0, β1, ... Therefore, for least squares analysis, the computational and inferential problems of polynomial regression can be completely addressed using the techniques of multiple regression. This is done by treating x, x2, ... as being distinct independent variables in a multiple regression model.

The goal of polynomial regression is to model a non-linear relationship between the independent and dependent variables (technically, between the independent variable and the conditional mean of the dependent variable).

In this report, we will explore the effect of different degrees of the polynomial regression model on the prediction results and whether increasing the degree will make the prediction value closer to the actual value.

1.2. Random forest regression

Concept

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.

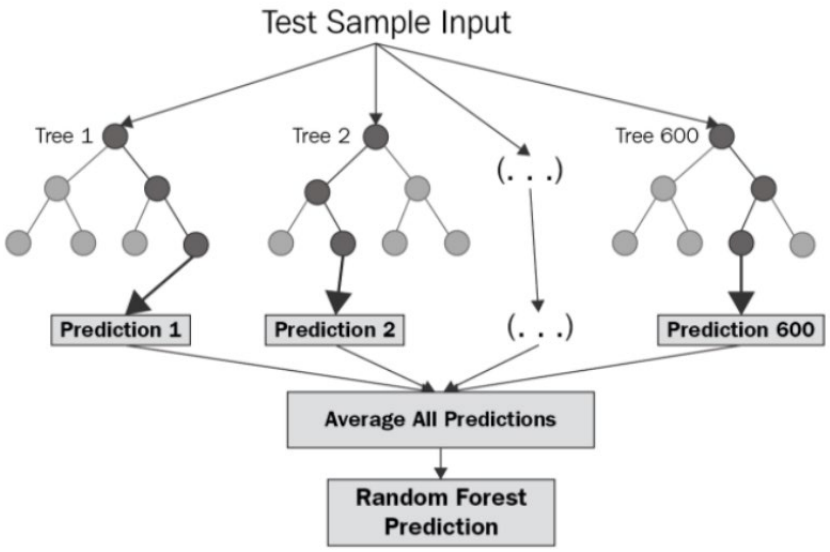


Figure 11: visualization of random forest algorithm

The diagram above shows the structure of a Random Forest. You can notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, overfitting may easily occur, we must choose the number of trees to include in the model.

In this report, we will use RandomizedSearchCV and GridSearchCV to filter out the best combination of hyperparameters, including hyperparameters like max_depth and min_samples_leaf, which are involved in decision tree pruning and n_estimators, to obtain the best model based on the training set while preventing overfitting. We will also explore the effect of different values of the same hyperparameters on the prediction results.

1.3. SVM regression

Concept

SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data. In contrast to OLS, the objective function of SVR is to minimize the coefficients — more specifically, the l2-norm of the coefficient vector — not the squared error. The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error, ϵ (epsilon).

SVMs can handle highly non-linear data using an amazing technique called kernel trick. It implicitly maps the input vectors to higher dimensional (adds more dimensions) feature spaces by the transformation which rearranges the dataset in such a way that it is linearly solvable. A kernel is a function which places a low dimensional plane to a higher dimensional space where it can be segmented using a plane. In other words, it transforms linearly inseparable data to separable data by adding more dimensions to it.

There are three kernels which SVM uses the most:

- Linear kernel: Dot product between two given observations
- Polynomial kernel: This allows curved lines in the input space
- Radial Basis Function (RBF): It creates complex regions within the feature spcae

In this report, we will explore linear kernel and RBF since polynomial kernel would lead to a curse of dimensionality. We will find the best-performing SVR model based on different kernel functions and other hyperparameters.

2. Classification model

We also intend to explore the classification model taught in class, but since the target of our dataset, SalePrice, is continuous data, we first need to classify it into low_price, medium_price, and high_price by different intervals before using the classification model. And then train the model using the training set to explore the performance of various classification models. Since our dataset is not a classification dataset, and we only roughly classify the target into three classes, so the next

part is only the result of our additional exploration of the original dataset. Next, we will introduce the classification models used in this report.

2.1. Naïve Bayes

Concept

Naïve Bayes classifier is a probabilistic machine learning model based on Bayes' theorem and conditional independence assumption that used for classification. With an equation:

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

We can find the probability of y happening, given that X has occurred. When y is the evidence and X is the hypothesis. In the equation, y is class variable and X is parameter or features.

$$X = (x_1, x_2, \dots, x_n)$$

and according to the chain rule we get.

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)}$$

For all entries in the dataset, the denominator does not change, it remains static. Therefore, the denominator can be removed, and a proportionality can be introduced.

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Then, we can find the predicted class value by the maximum probability.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

In this report, there are three Naïve Bayes that used to find predicted class. Gaussian Naïve Bayes, when the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution. Second is Multinomial Naïve Bayes, which is mostly

used for document classification problem. The last one is Complement Naïve Bayes. This method calculates the probability of the item belonging to all the classes.

2.2. Random forest classification

Concept

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. Uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions.

The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

Model performance

Polynomial Regression

According to the previous analysis, we are not going to use PCA in the regression algorithm part because it will degrade the final performance as long as it is applied. First, we use polynomial regression on the standardized dataset, we have tried to set the degree as a positive integer from 1 to 10, but we found that when degree is greater than or equal to 4, it will lead to Memory Error because of curse of dimensionality, so we only explore the case when degree is equal to 1, 2, and 3.

```
MemoryError                               Traceback (most recent call last)
d:\PythonProjects\DATA7703\Project\Jia_poly_reg.ipynb Cell 4 in <cell line: 3>()
      3 for i in degrees:
      4     poly_reg = PolynomialFeatures(degree=i, include_bias=False)
----> 5     X_poly = poly_reg.fit_transform(X_train)
      6     lin_reg = LinearRegression()
      7     lin_reg.fit(X_poly, y_train)

File d:\Python3\lib\site-packages\sklearn\base.py:867, in TransformerMixin.fit_transform(self, X, y, **fit_params)
    863 # non-optimized default implementation; override when a better
    864 # method is possible for a given clustering algorithm
    865 if y is None:
    866     # fit method of arity 1 (unsupervised transformation)
--> 867     return self.fit(X, **fit_params).transform(X)
    868 else:
    869     # fit method of arity 2 (supervised transformation)
    870     return self.fit(X, y, **fit_params).transform(X)

File d:\Python3\lib\site-packages\sklearn\preprocessing\_polynomial.py:432, in PolynomialFeatures.transform(self, X)
    428     XP = sparse.hstack(columns, dtype=X.dtype).tocsc()
    429 else:
    430     # Do as if _min_degree = 0 and cut down array after the
    431     # computation, i.e. use _n_out_full instead of n_output_features_.
--> 432     XP = np.empty(
...
    448
    449     # degree 0 term
    450     if self.include_bias:

MemoryError: Unable to allocate 209. GiB for an array with shape (1168, 24040015) and data type float64
```

Figure 12: Memory Error when degrees larger than 3

Since our test set does not have an exact target value, we use train_test_split in sklearn.model_selection to split the standardized dataset into a training set and a test set, and begin our exploration.

```
Degree 1 train MSE: 1016302280.244
Degree 1 test MSE: 16859952839972054309384856430182 4.000
```

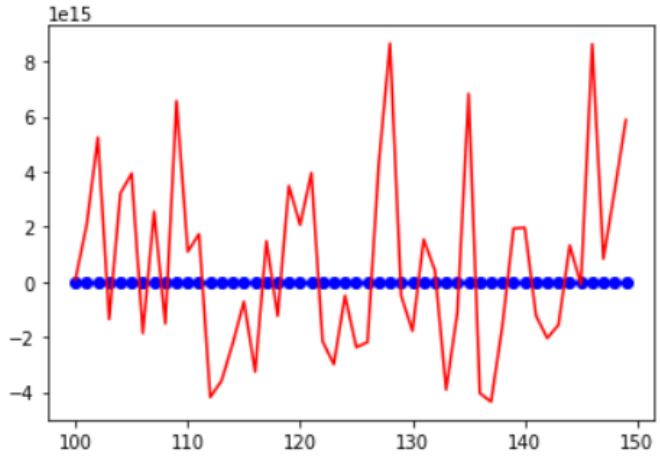


Figure 13: predicted and actual SalePrice when degree is 1

The blue points in the figure represent the true value of the SalePrice from the 100th to the 150th index of the test set, and the red line represents the predicted Sale Price of the current model.

```
Degree 2 train MSE: 0.000
Degree 2 test MSE: 3314311808.758
```

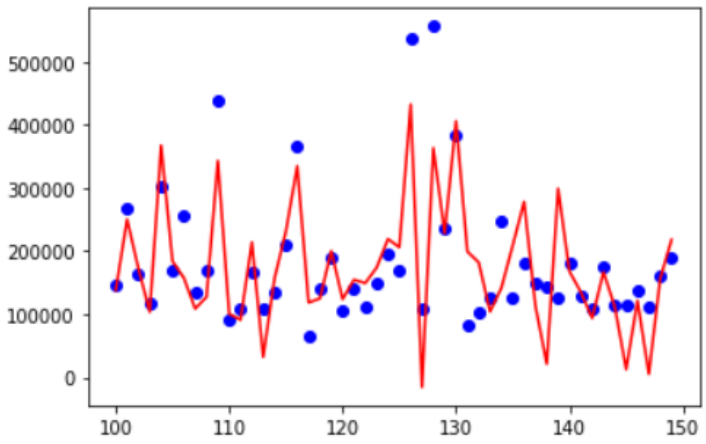


Figure 14: predicted and actual SalePrice when degree is 2

```
Degree 3 train MSE: 0.000
Degree 3 test MSE: 1091503574.482
```

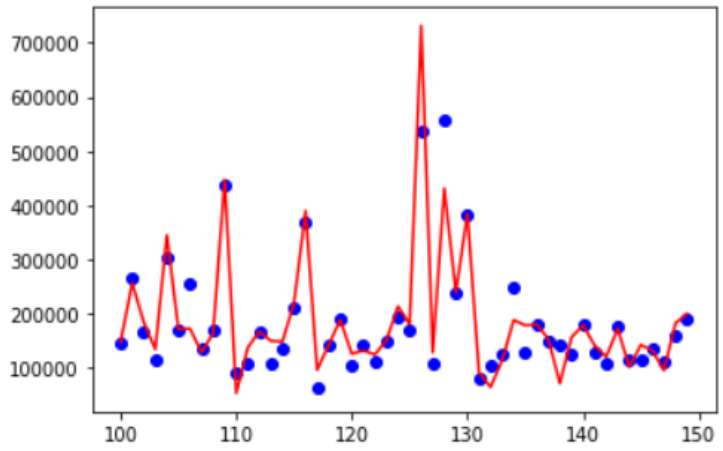


Figure 15: predicted and actual SalePrice when degree is 3

We found that when degree is equal to 2 and 3, the training set MSE are both 0, which means that this model is overfitting, probably due to too many attributes in our model; the MSE of the test set is gradually decreasing as the complexity of the model rises.

SVM Regression

Then we apply Support vector regression on the standardized dataset to explore whether there will be better results. Compared to polynomial regression, Support vector regression requires tuning three hyperparameters. Our aim is to improve the model in any way possible. One important factor in the performances of these models are their hyperparameters, once we set appropriate values for these hyperparameters, the performance of a model can improve significantly.

In this project, we would use GridSearchCV in sklearn.model_selection to find the best combination. GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function, we get accuracy/loss for every combination of hyperparameters, and we can use the best_params_ attribute to get the best performance. We put the following hyperparameter combinations into GridSearchCV:

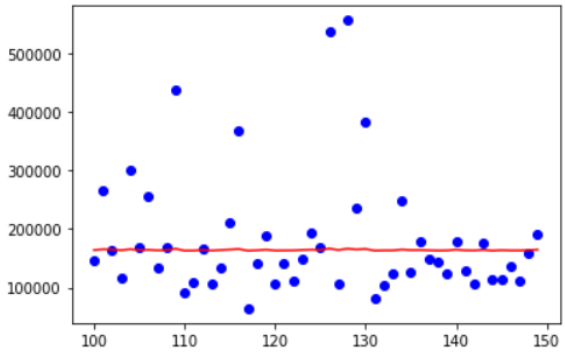
```
{ 'C': [int(x) for x in np.linspace(start=10, stop=10000, num=101)],
```

'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

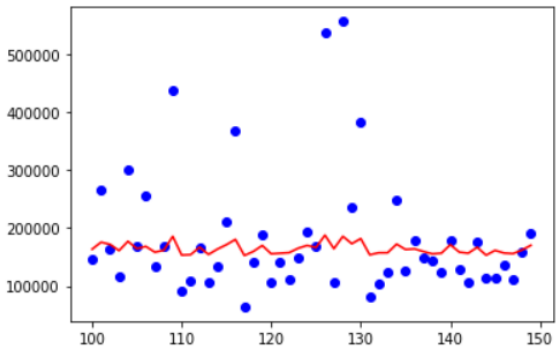
'kernel': ['rbf','sigmoid'] }

And then got the best hyperparameter combination as SVR(C=10000, gamma=0.01, kernel='sigmoid'), based on which, in order to explore the effect of different parameters on the prediction results, I kept the parameters except Regularization parameter constant and only changed the value of it to see the performance difference between the predicted values.

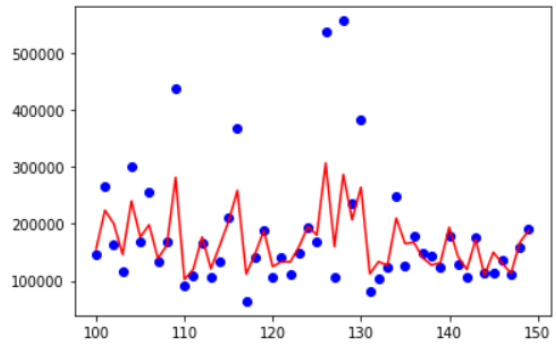
C=10 train MSE: 6512699648.489
C=10 test MSE: 6245114559.021



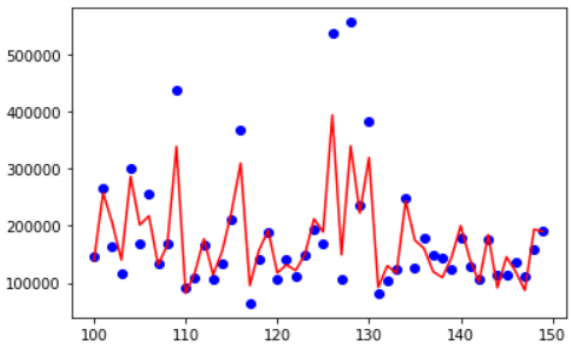
C=100 train MSE: 5510602335.215
C=100 test MSE: 5225879524.203



C=1000 train MSE: 2111170296.203
C=1000 test MSE: 1754128887.352



C=10000 train MSE: 1283227814.635
C=10000 test MSE: 884627592.799



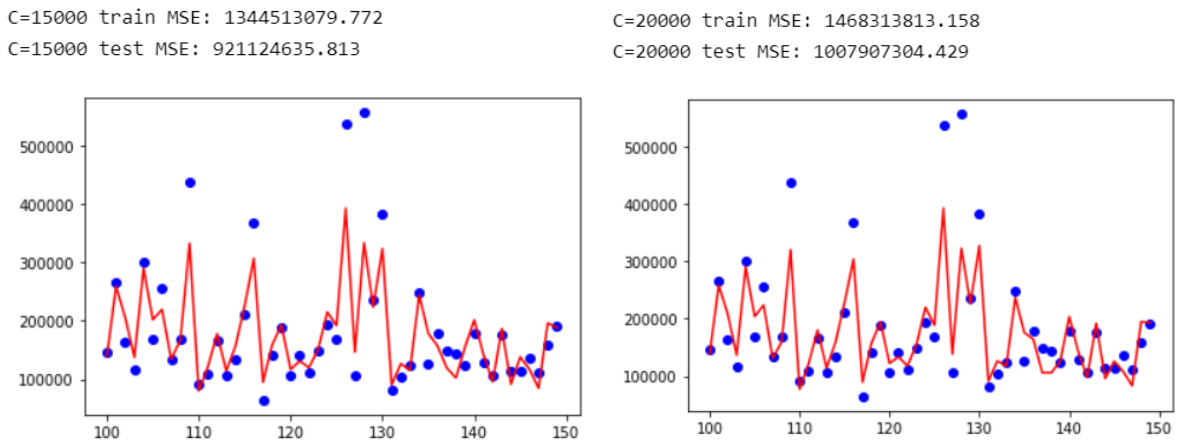


Figure 16: How different regularization parameter changes the model

We can conclude that the MSE becomes better as the regularization parameter increases within a fixed model complexity interval, because the strength of the regularization decreases. However, when the regularization parameter exceeds 10,000, the MSE of the test set will increase, which is in line with the rule that the error of the test set decreases first and then increases as the complexity of the model increases. The MSE of the training set fluctuates with the increase of the complexity of the model and does not decrease all the time.

Random Forest Regression

Finally, we applied random forest regression on the standardized dataset to see how it performs. Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster; compared to support vector regression, random forest regression has more hyperparameters, so I used RandomizedSearchCV before GridSearchCV. In contrast to GridSearchCV, not all parameter values are tried out; instead, a fixed number (in our project: 200) of parameter settings is sampled from the specified distributions. We use it to narrow down the range of hyperparameters and save the time of GridsearchCV. We put the following hyperparameter combinations into RandomizedSearchCV:

{ 'n_estimators': [int(x) for x in np.linspace(start=100, stop=400, num=11)],
‘max_features’: [‘auto’, ‘sqrt’],
‘min_samples_leaf’: [1, 2, 3, 4, 5],
‘min_samples_split’: [1, 2, 3, 4, 5],

‘max_depth’: [int(x) for x in np.linspace(10, 110, num=11)]

‘bootstrap’: [True, False] }

And then the best hyperparameter combination with RandomizedSearchCV is:

```
{ 'n_estimators': 370,
  'min_samples_split': 3,
  'min_samples_leaf': 1,
  'max_features': 'sqrt',
  'max_depth': 30,
  'bootstrap': False}
```

On top of these parameters, we then put the following hyperparameter combinations into GridSearchCV:

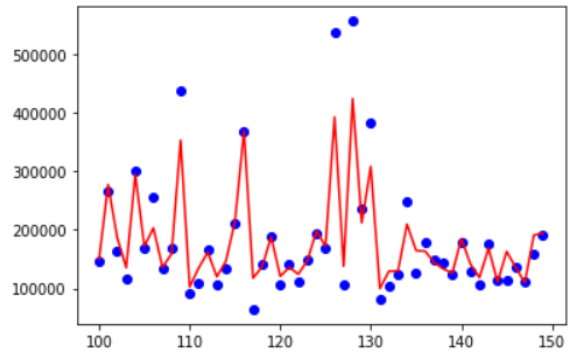
```
{ 'bootstrap': [False],
  'max_depth': [26, 27, 28, 29, 30, 31, 32],
  'max_features': ['sqrt'],
  'min_samples_leaf': [1, 2],
  'min_samples_split': [2, 3, 4],
  'n_estimators': [365, 370, 375]}
```

The best combination of hyperparameters after GridSearchCV is listed as below:

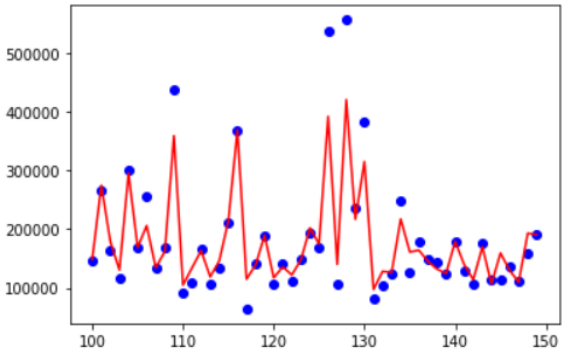
```
{ 'n_estimators': 370,
  'min_samples_split': 3,
  'min_samples_leaf': 1,
  'max_features': 'sqrt',
  'max_depth': 28,
  'bootstrap': False}
```

This is basically consistent with the results of RandomizedSearchCV, so we then explore the effect of different values of max_depth on the prediction results. I kept the parameters except max_depth parameter constant and only changed the value of it to see the performance difference between the predicted values.

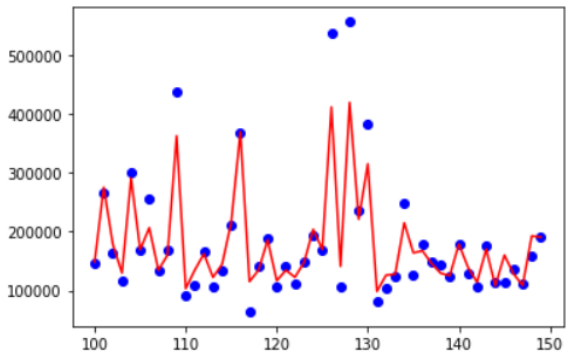
max_depth=10 train MSE: 59556419.544
max_depth=10 test MSE: 746209141.090



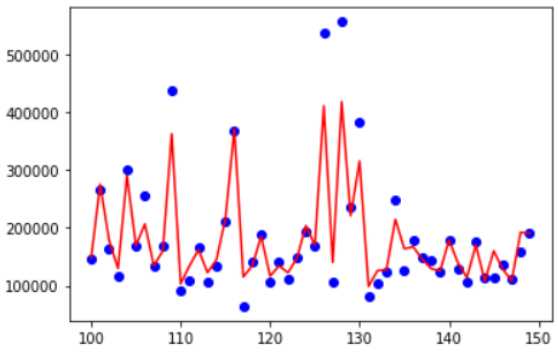
max_depth=20 train MSE: 3107147.041
max_depth=20 test MSE: 740156764.445



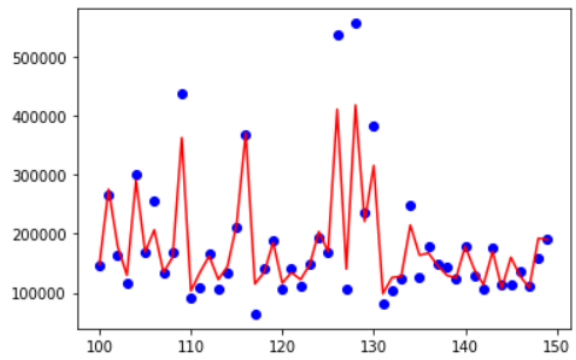
max_depth=28 train MSE: 3091626.333
max_depth=28 test MSE: 692126172.632



max_depth=35 train MSE: 3136920.045
max_depth=35 test MSE: 695308761.147



max_depth=40 train MSE: 3136920.045
max_depth=40 test MSE: 695308761.147



max_depth=50 train MSE: 3136920.045
max_depth=50 test MSE: 695308761.147

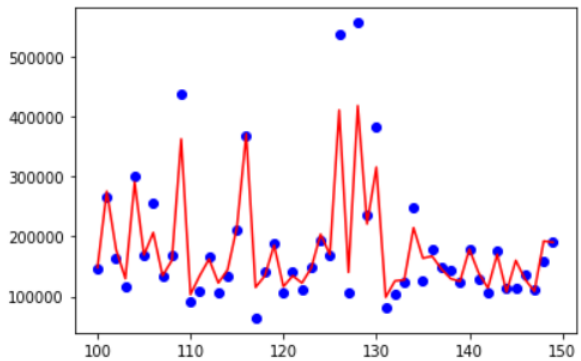


Figure 17: How different max_depth parameter changes the model

We found that when max_depth was greater than 35, the MSE of both the test set and the training set were unchanged, indicating that we did not have over-fitting in the tree generation phase because these pruning settings did not make the MSE of the test set better.

Finally, we explore the features that have the greatest impact on the model by using the feature_importances_ attribute of RandomForestRegressor.

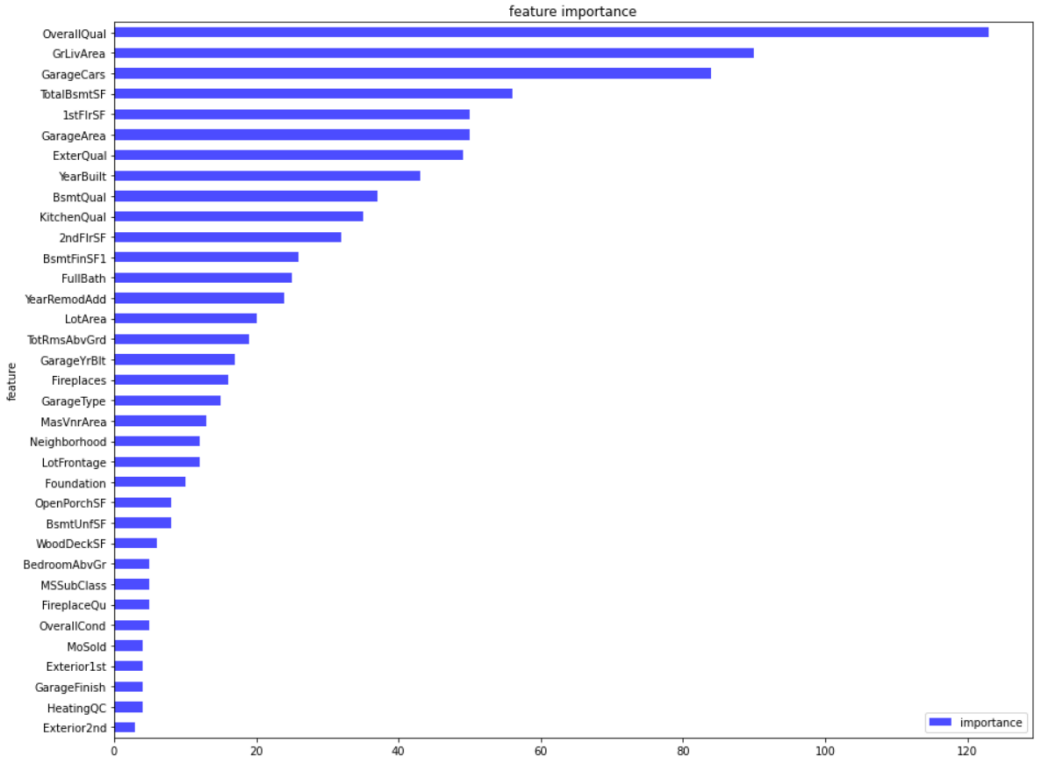


Figure 18: Feature importance of random forest regression model

This graph shows the impurity-based feature importance of the first 35 features of our best random forest model in ascending order, and we find that in addition to overall quality, Above ground living area square feet, Size of garage in car capacity, and Total square feet of basement area has the most significant impact on house price, probably because these parameters can reflect the total area of the house from the side.

To conclude, comparing the MSEs of random forest regression with polynomial regression and SVM regression, random forest regression has the best performance on this dataset.

Naïve Bayes

According to our output, output is a regression output we cannot apply classification to our current dataset. So, before we apply this algorithm, we divide our output into three classed which are low price, medium price and high price.

If sale price is less than 129975, it belongs to low price class. If sale price is higher than 129975 but less than 214000, this range belongs to medium price. For high price class, sale price is higher than 214000.

1. Gaussian Naïve Bayes

   The result of this model prediction is in the table below.

Table 3: result of Gaussian Naïve Bayes prediction

	precision	recall	f1-score	support
1	0.60	0.89	0.72	114
2	0.81	0.57	0.67	220
3	0.74	0.84	0.79	104
accuracy			0.71	438
macro avg	0.72	0.76	0.72	438
weighted avg	0.74	0.71	0.71	438

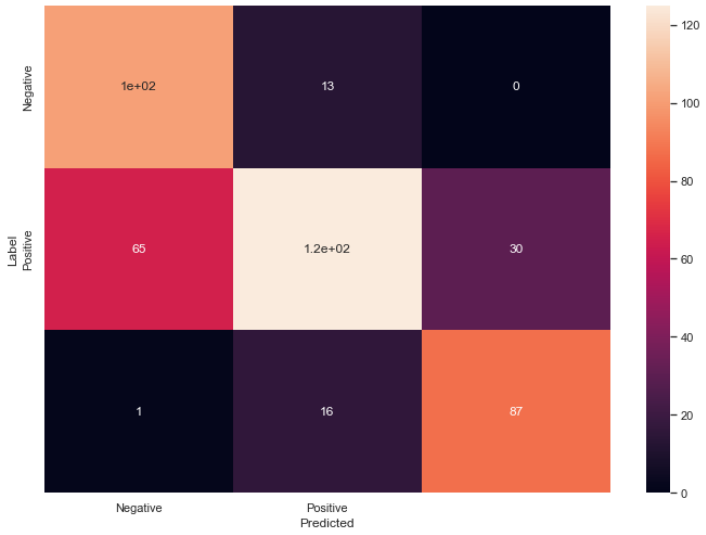


Figure 19: confusion matrix of Gaussian Naïve Bayes model

2. Multinomial Naïve Bayes

The result of this model prediction is in the table below.

Table 4: result of Multinomial Naïve Bayes prediction

	precision	recall	f1-score	support
1	0.56	0.84	0.67	114
2	0.65	0.50	0.57	220
3	0.60	0.56	0.58	104
accuracy			0.61	438
macro avg	0.60	0.63	0.61	438
weighted avg	0.62	0.61	0.60	438

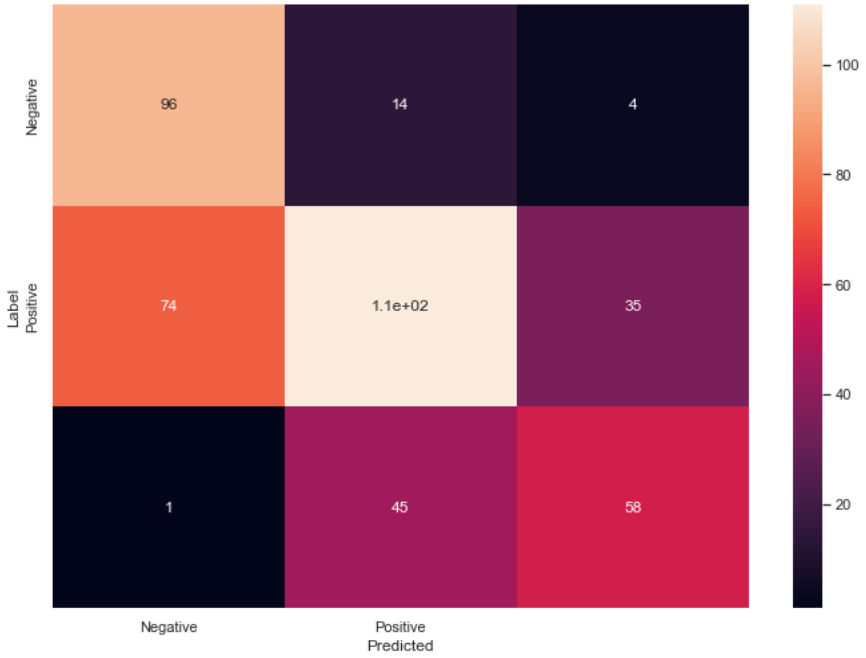


Figure 20: confusion matrix of Multinomial Naïve Bayes model

3. Complement Naïve Bayes

The result of this model prediction is in the table below.

Table 5: result of Complement Naïve Bayes prediction

	precision	recall	f1-score	support
1	0.46	0.78	0.58	114
2	0.60	0.26	0.37	220
3	0.54	0.78	0.64	104
accuracy			0.52	438
macro avg	0.54	0.61	0.53	438
weighted avg	0.55	0.52	0.49	438

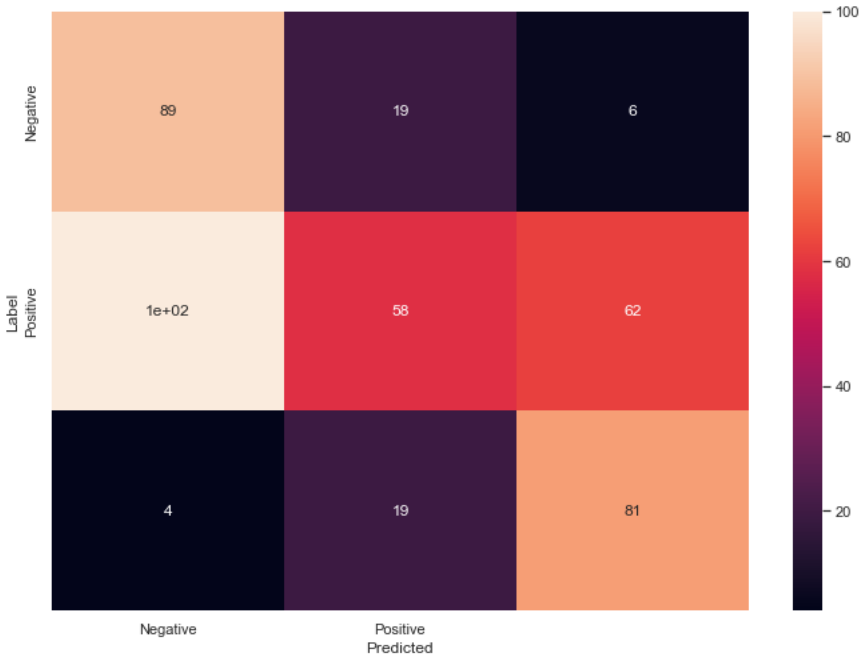


Figure 21: confusion matrix of Complement Naïve Bayes model

Finally, we compare the result of these three algorithms which is the graph below.

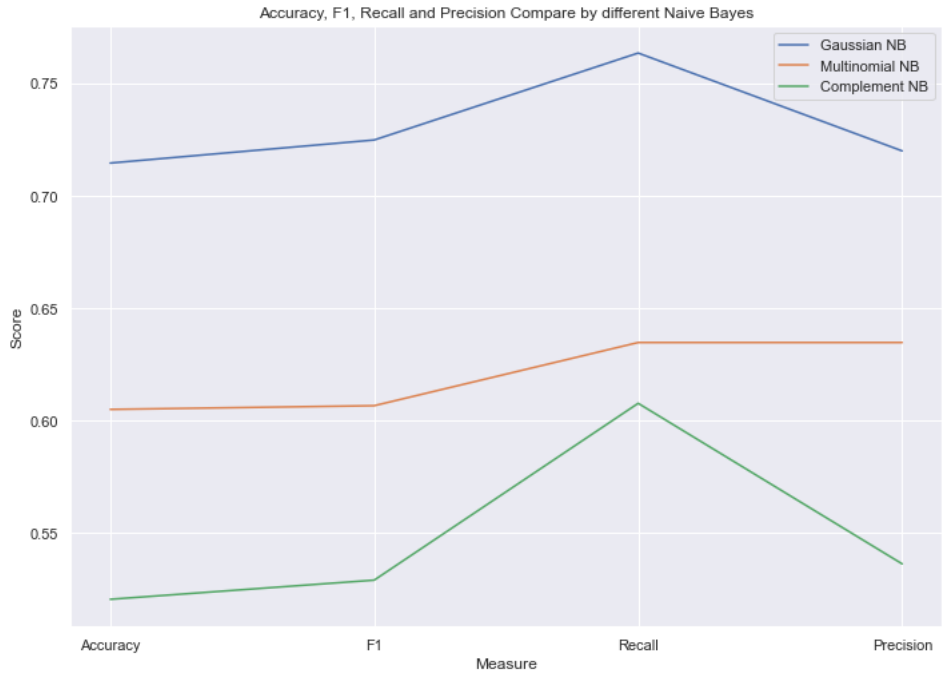


Figure 22: Accuracy, F1, Recall and Precision of each Naïve Bayes model

We can see that Gaussian Naïve Bayes has the best performance among three models.

Random Forest Classification

Before applying this algorithm, I set the house price as less than 25% as low_price, 25% to 75% as medium_price, and higher than 75% as high_price.

The idea of tuning hyperparameters is basically the same as random forest regression, we use RandomizedSearchCV first and then GridSearchCV, the best set of hyperparameters is ：

{ 'n_estimators': 180,
'min_samples_split': 2,
'min_samples_leaf': 1,
'max_features': 'sqrt',
'criterion': 'gini',
'max_depth': 61,
'bootstrap': False}

The accuracy and f1_score of the test set is shown below. Such a high f1_score may be because we just simply divide the target into three classes.

```
test accuracy:  0.9931506849315068
test f1_score:  0.9931506849315068
```

We then explore the effect of different values of max_depth on the prediction results. I kept the parameters except max_depth parameter constant and only changed the value of it to see the performance difference between the predicted values.

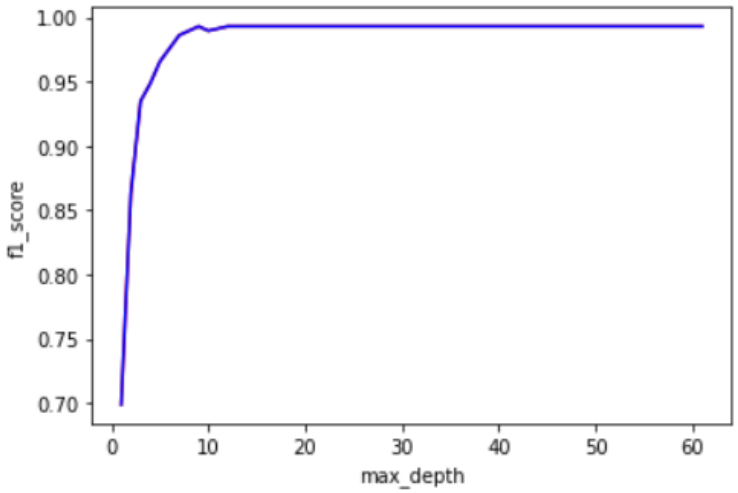


Figure 23: Curve of f1_score of the test set as max_depth changes

We found that when max_depth is greater than or equal to 12, the f1_score has been maintained at 0.993151; this is different from the GridsearchCV results, probably because the division of the test set and training set during GridsearchCV is different from the model testing phase

Finally, we explore the features that have the greatest impact on the model by using the feature_importances_ attribute of RandomForestClassifier.

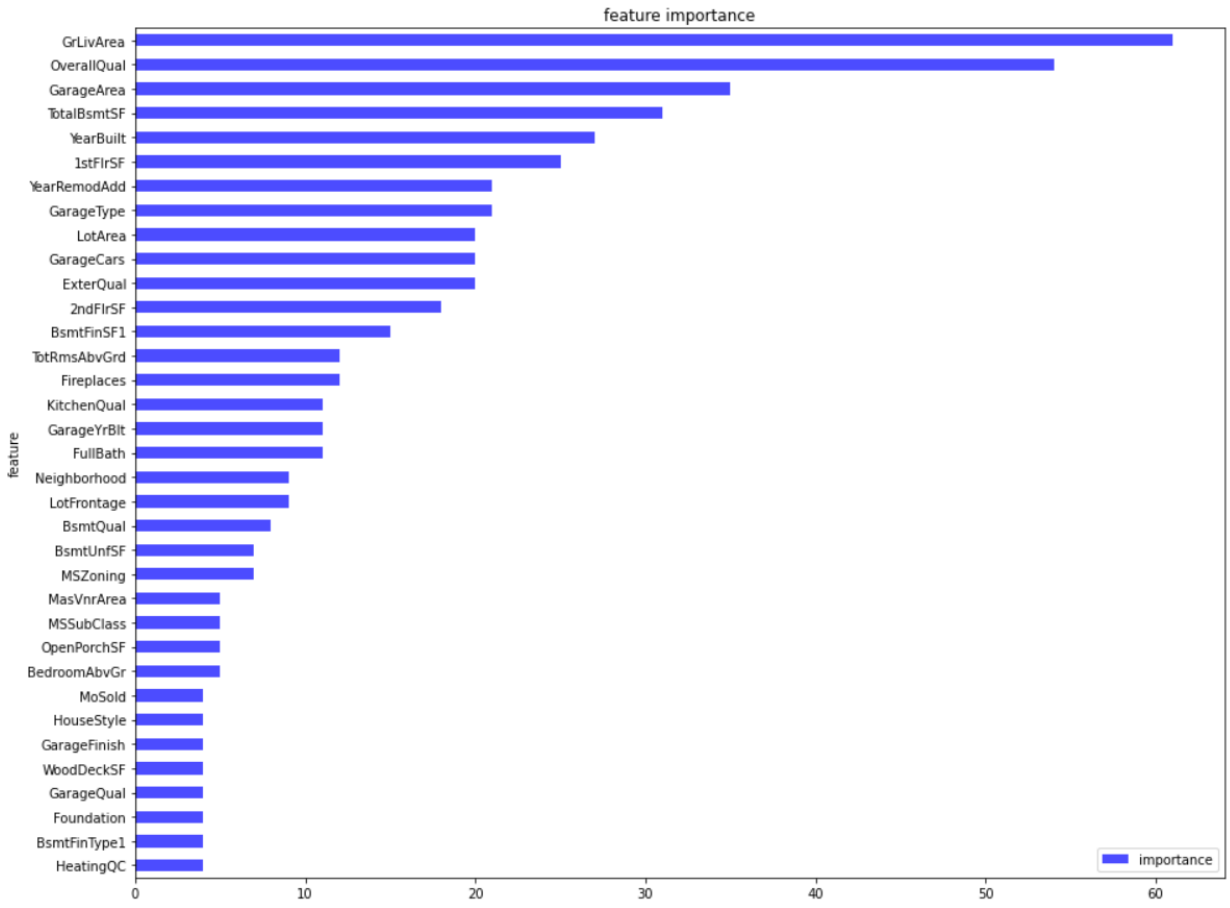


Figure 24: Feature importance of random forest regression model

This graph shows the impurity-based feature importance of the first 35 features of our best random forest classification model in ascending order, and we find that Above ground living area square feet becomes the most influential feature of the model, which is different from the results of random forest regression one, and other features like Year Built have increased in importance in classification model.

Conclusion & Future studies

Data preprocessing, we used deterministic regression imputation for imputing missing data. However, even though the average cross-validation is quite high but there are still some concerns. If we look at the figure 3 of Electrical and SalePrice data, we can see that the imputed value is not close to the rest of the data and same as MasVnrType. Another way to improve this problem is stochastic regression imputation, stochastic regression imputation will reduce bias and it could be done by doing extra step of augmenting each predicted score with a residual term. However, it may not fit for our dataset. If we would like to use it, it would be better to compare to other imputation methods by using cross-validation. Another thing to mention is, encoding method, in this report we used label encoding for encoding our data. However, one-hot encoding would probably be better because one-hot encoding only uses 0 and 1 and this method can assure that the model will not assume that the higher numbers are more important as label encoding. Nevertheless, when we compare one-hot encoding to label encoding with our dataset, as you can see the result from above, it turned out that the average cross-validation score is quite similar. This implies that we could use both methods with our dataset. Even though one-hot encoding would probably be better but there are too many columns after using one-hot encoding, for example our dataset there are columns more than 900 columns. This would lead to a bad execution and use a lot of time to execute.

We showed that applying PCA in this dataset is not a good choice because as soon as it is used, the performance of the regression model decreases, which is not what we expected at the beginning because there are some positive correlations between our features from the dataset, such as First-floor square feet and Second-floor square feet. Still, the advantage of PCA is that the model's training time decreases after using it, and the memory error due to the curse of dimensionality is not encountered as in polynomial regression. The performance of regression algorithms increases with more PCA components (with a token selection of hyperparameters) and is consistent with objective facts if only the application of PCA is compared.

Our project also has a lot of limitations, which can be roughly divided into two parts; the first is that we found that data preprocessing is a rather important part, for example, when the

hyperparameters of our regression model have been tuned to close to the best, the time spent modifying the hyperparameters can actually bring minimal improvement. Suppose we still want to improve model performance. In that case, we have to spend more time on the preprocessing of the dataset, such as exploring various combinations of encoding and imputation, plus outlier detection. More importantly, during the EDA process, we found that many features strongly correlate with each other. So, if we have more time, we will try to reduce the dimensionality of these features individually to make the model achieve better results.

In addition, there is another limitation that our project can improve: we can try to use more regression models, relatively speaking we only used two basic regression models in this project and one ensemble method of bagging; we can spend more time trying boosting, such as AdaBoost, XGBoost, this will also allow us to explore the differences between bagging and boosting at the same time. In addition, we can try to use a neural network to explore whether it will have better results, and we can use Pytorch to help us achieve it better.

Reference

[1] "Imputation (statistics)", (2022). https://en.wikipedia.org/wiki/Imputation_(statistics)

[2] Jakobsen, J.C., Gluud, C., Wetterslev, J. et al. "When and how should multiple imputation be used for handling missing data in randomised clinical trials – a practical guide with flowcharts". BMC Med Res Methodol 17, 162 (2017). https://doi.org/10.1186/s12874-017-0442-1https://www.missingdata.nl/missing-data/missing-data-methods/imputation-methods/

[3] Rohith Gandhi, "Naïve Bayes Classifier",(2018). https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[4] Iris Eekhout, "Imputation Methods", (2017). https://www.missingdata.nl/missing-data/missing-data-methods/imputation-methods/

[5] Geeksforgeeks, elbow method https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

[6] chaya. (2020, June 9). Random Forest Regression. gitconnected. Retrieved November 1, 2022, from https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

[7] Sethi, A. (2020, April 1). Support Vector Regression Tutorial for Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/

[8] Wikipedia contributors. (2022, October 3). Polynomial regression. Wikipedia. https://en.wikipedia.org/wiki/Polynomial_regression