COMP SCI 7209 Big Data Analysis and Project
Project Title: Big Data: House Price Analysis and Prediction (Part C)
Ning Ni (a1869549)
Project Coordinator: Bernard Evans
20 July 2023

1 Problem Description

1.1 Introduction

The purpose of this project is to leverage house features to construct one or multiple models for predicting house prices and price ranges.

1.2 Data Description

Data source from Dean De Cock (2011, p.4). It comprises 2,930 observations and contains 80 explanatory variables, including 23 nominal, 23 ordinal, 14 discrete, and 20 continuous variables.

1.2.1 Input Data

Input dataset were categorized into four classes: Structure and Layout, Location and Geography, Material and Quality, Facilities.Through feature engineering, 29 variables were selected as the predictors including 22 categorical variables and 7 numerical variables. After preprocessing, the input data consists of 128 dummy variables and 7 numeric variables, totaling 135 variables to meet the requirements of the input data format of some models.

Table 1 Summary of Input Data

Attribution	Variable Type	Variable Name	Count
Structure and Layout	Cat*	HouseStyle, GarageType	9
	Num**	LotArea, YearBuilt, GarAreaPerCar, TotalHouseSF, GrLivAreaPerRoom, TotalFullBath, TotalPorchSF	
Location and Geography	Cat	LotShape, MSZoning, PavedDrive	3
	Num	None	
Material and Quality	Cat	OverallQual, OverallCond, Exterior 1st, MasVnrType, Foundation, BsmtQual, KitchenQual, GarageQual, IsRemodGar, IsRemod	11
	Num	MasVnrArea	
Facilities	Cat	HeatingQC, CentralAir, FireplaceQu, SaleType, SaleCondition	6
	Num	Fireplaces	

*Categorical **Numerical

1.2.2 Output Data

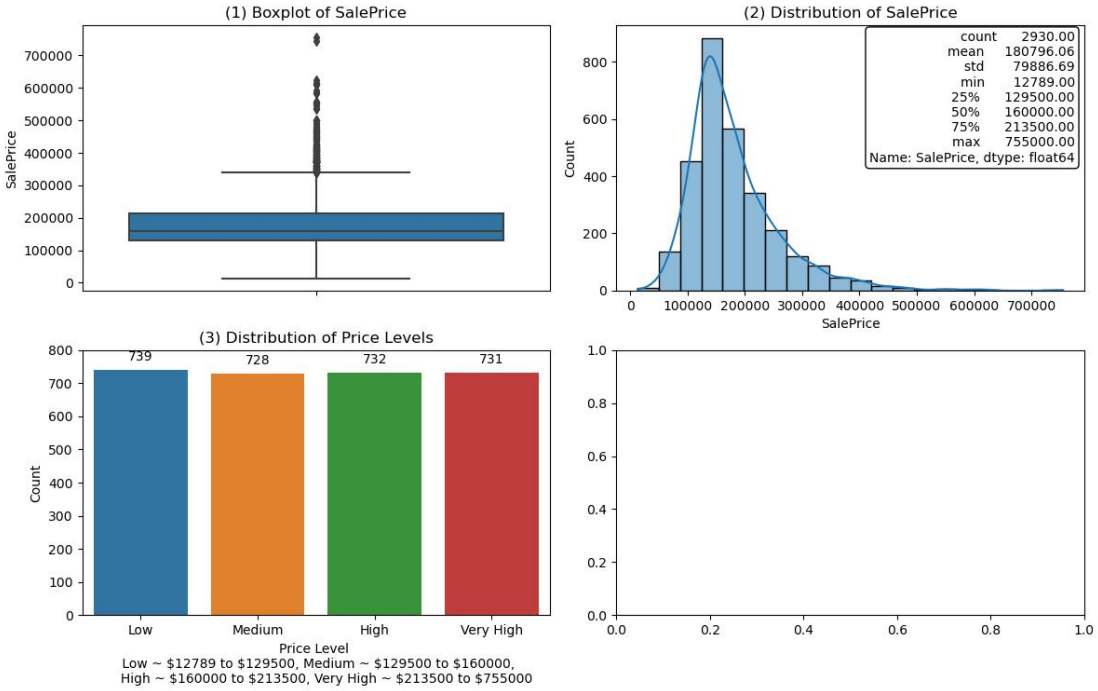
The task of predicting house price is a regression problem, where the output is a numeric variable represented as a float type.While predicting house price range is treated as a categorical problem, where the output is a categorical variable with four levels: low, medium, high, and very high.

Table 2 Summary of Output Data

Output Name	Problem Type	Output Type	Output Format
House Price	Regression	Num**	Float
Price Range	Classification	Cat*	"Low", "Medium", "High", "Very High"

*Categorical **Numerical

Figure 1: Analysis of Target Variable



2 Data Pre-processing

2.1 Imputation

Downey et al. (2015) provided methods for imputing missing values. The missing values will be filled with the values "None" or "0". This indicates that some houses lack relevant feature information.

2.2 Scaling

Ahsan et al. (2021,p.52) stated the principle of feature scaling is to transform feature values of different scales into a similar scale, avoiding the bias of machine learning models towards certain features due to their varying scales. In this project, the chosen method for numerical feature scaling is Standardization, also known as the Z-score method. Converting categorical variables into "one-hot" encoding.

2.3 Pre-processing Transformations

2.3.1 Feature Removal

- Removing features that are irrelevant to the response variable, such as "PID" and "Order."
- Removing the observation with obvious outliers like houses with pricing in only hundreds of dollars.
- Removing features that have less correlation with the response variable. This project employs correlation coefficients and chi-square tests, using reasonable thresholds to assess the degree of correlation (detailed in Part B 3.4).
- Rmoving features with collinearity to eliminate redundant information among features. The method used is similar as mentioned in the previous point.

2.3.2 Feature Aggregation

Galli (2022, p.217) provided several approaches to combine feature. In this project, methods of mathematical functions and reference variables were employed to aggregate features that may contain correlated information into a new feature. For example, calculate the mean area of rooms by dividing the total area of all rooms by the number of rooms.

2.4 The Result of Pre-processing

Table 3 The Result of Pre-processing

Items	Feature Name	Count
Deleted numeric features	'LotFrontage','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','2ndFlrSF','LowQualFinSF','BsmtFullBath','BsmtHalfBath', 'HalfBath','BedroomAbvGr','KitchenAbvGr', 'YrSold', 'WoodDeckSF', 'EnclosedPorch', 'MoSold', 'PID', 'Order', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'GarageYrBlt','GarageArea','GarageCars','GrLivArea', 'FullBath','TotRmsAbvGrd','TotalBsmtSF','1stFlrSF', 'OpenPorchSF','YearRemod/Add'	29
Deleted categorical features	'BsmtCond','Electrical','Fence','Alley','Condition1', 'BsmtFinType2','BldgType','RoofStyle','LandContour', 'ExterCond','Functional','LotConfig','Heating','Condition2','Street','RoofMatl','LandSlope','MiscFeature','PoolQC', 'Utilities','MSSubClass','Exterior2nd','GarageFinish', 'ExterQual','Neighborhood','BsmtExposure', 'BsmtFinType1','GarageCond'	28
Reserved numerical features	'LotArea','YearBuilt','MasVnrArea','Fireplaces'	4
New built numerical features	'GarAreaPerCar': the garage area per car, 'TotalHouseSF': the total floor area of a house, GrLivAreaPerRoom':The average area per room on the ground floor, 'TotalFullBath': the total number of full bathroom, 'TotalPorchSF': the total area of porch, 'IsRemodGar': If the garage is built after the house is constructed, 'IsRemod': Whether the house has been renovated.	7
Reserved categorical features	'MSZoning','LotShape','HouseStyle','OverallQual','OverallCond','Exterior1st','MasVnrType','Foundation', 'BsmtQual','HeatingQC','CentralAir','KitchenQual', 'FireplaceQu','GarageType','GarageQual','PavedDrive', 'SaleType','SaleCondition'	18

3 Model Selection

3.1 Selection of Regression Model

Linear Regression, Random Forest, Adaboost and XGBoost are selected as candidate models to solve the house price prediction problem. Compatibility with high-dimensional input data is an important selection criterion.

- Linear Regression
  Easy to implement and interpret. The model structure is intuitive and can clearly reveal the linear relationship between house features and price.
- Random Forest
  Capable of automatically handling non-linear relationships, no need for feature scaling, and has strong capability to handle high-dimensional data.
- Adaboost
  Schapire (2013, p.37) stated Adaboost is an intuitive and simple ensemble learning algorithm with adaptability, suitable for small-sized datasets with high-dimensional features in this project.
- XGBoost
  Avanijaa (2021, p.2151) mentioned that with high robustness and accuracy, XGBoost is well-suited for large-scale datasets. Despite the input data containing only 2930 samples, XGBoost can be introduced into this project for exploration and comparison with other models.

3.2 Selection of Classification Model

Naive Bayes, SVM, Random Forest, Adaboost are selected as candidate models to solve the price range classification problem.

- Naive Bayes
  A fast and simple classification algorithm that is easy to implement. It's suitable for handling discrete features.Considering the input data contains as many as 128 dummy variables, it may be a suitable model.
- SVM
  SVM can handle non-linear classification problems through kernel functions, and it is known for achieving highly accurate predictions in classification tasks.
- Random Forest and Adaboost
  Same as regression problem mentioned above.

3.3 Summary of Model Selection

Table 4 Summary of Model Selection

Problem Type	Target Variable	Model Selection	Performance Evaluation	Validation Method
Regression	House Price ("SalePrice")	Linear Regression Random Forest Adaboost Xgboost	RMSE MAE R-square	Cross-Validation
Classification	Price Range ("PriceLevel")	Naive Bayes Support Vector Machine Random Forest Adaboost	Accuracy Recall AUC	Cross-Validation and/or Bootstrap

4 Model Refinement and Performance Description

4.1 Regression Problem

4.1.1 Linear Regression

Table 5 Refinement Step of Linear Regression Model

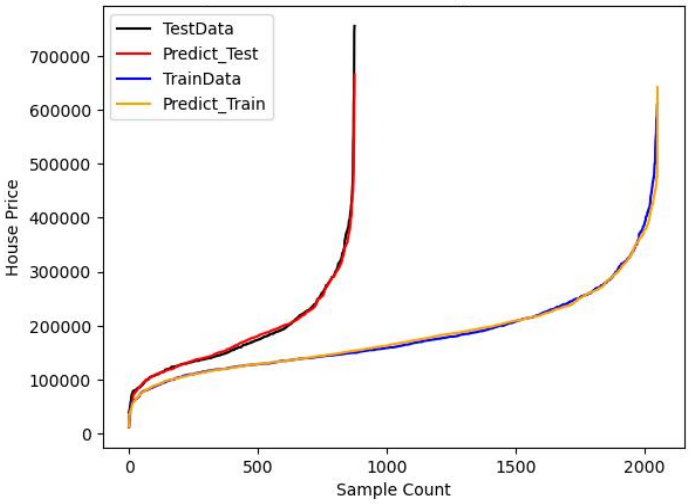
Order	Refinement Step	Details
1	Feature Engineering	● Standardized numeric variables ● Created dummy variables for categorical variables. ● Target Variable is SalePrice
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.linear_model import LinearRegression ● from sklearn.linear_model import Ridge
4	Hyper-parameters Adjusting	● alpha in Ridge Regression
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● RMSE Root Mean Squared Error emphasizes the closeness between predicted values and true values, and a smaller RMSE indicates a more accurate model prediction ● MAE Mean Absolute Error focuses on the absolute differences between predicted values and true values, and a smaller MAE indicates a more accurate model prediction. ● R2 The coefficient of determination (R2) is used to measure the extent to which the model explains the variability of the target variable (house prices), and its values range from 0 to 1. When R2 is close to 1, it indicates that the model explains the variability of house prices better, resulting in a better fit and stronger predictive ability.

4.1.1.1 Linear Regression Model

Table 6 Evaluation of Linear Regression Model

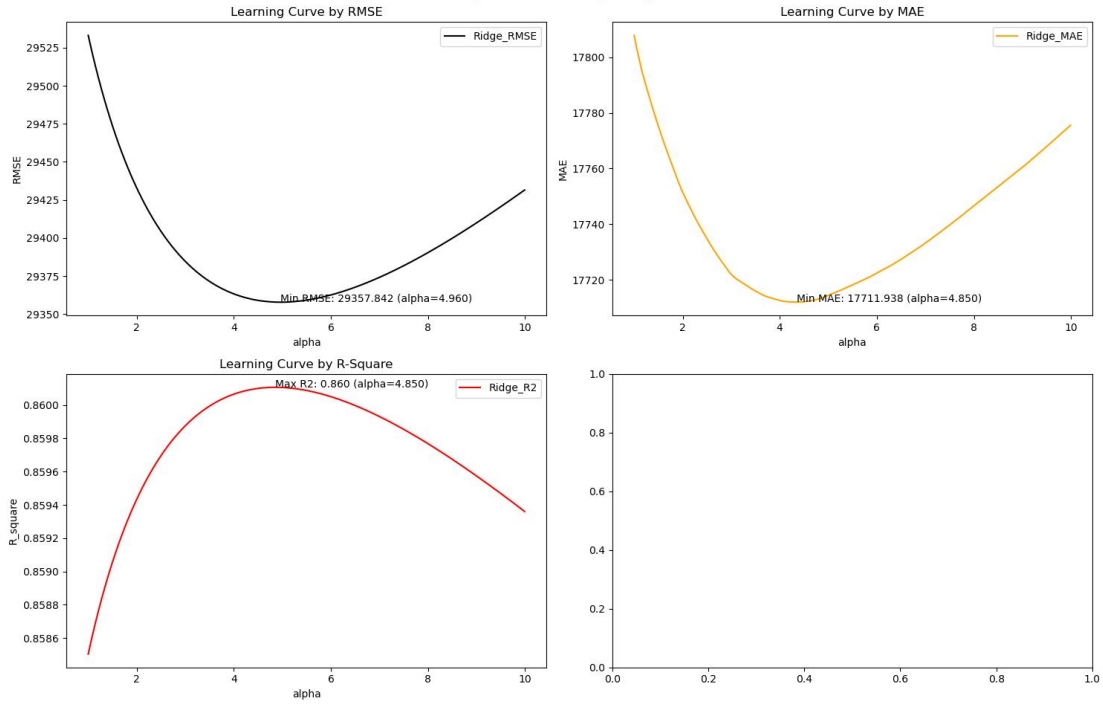
Model	Data Set	RMSE	MAE	R2
Linear Regression	Training	25915.02	16411.02	0.891
	Testing	35802.52	19490.85	0.815

Figure 2 Fit Plot of Linear Regression Model



4.1.1.2 Ridge Regression Model

Figure 3 Learning Curve for Ridge Regression Model



alpha = 4.85 is the best parameters in the cross-validation of training dataset

Table 7 Evaluation of Ridge Regression Model

Model	Data Set	RMSE	MAE	R2
Ridge Regression	Training	29357.84	17711.94	0.860
	Testing	35532.72	19330.66	0.818

4.1.2 Random Forest

Table 8 Refinement Step of Random Forest Regression Model

Order	Refinement Step	Details
1	Feature Engineering	● Label encoded categorical variables ● Target Variable is SalePrice
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.ensemble import RandomForestRegressor
4	Hyper-parameters Adjusting	● 'n_estimators':[*range(50,501,50)] ● 'max_depth':[*range(5,20,2)] ● 'min_samples_leaf':[*range(1,10,2)] ● 'min_samples_split':[*range(1,6)] ● Fine tuning after Coarse tuning
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● RMSE ● MAE ● R2

In coarse tuning，{'max_depth': 15, 'n_estimators': 300, 'min_samples_leaf': 1, 'min_samples_split': 4} were the optimal parameters. The Fine tuning was shown as Figure 4. The final parameter 'n_estimators' :280, max_depth' :15

Figure 4 Learning Curve for Random Forest Regression Model

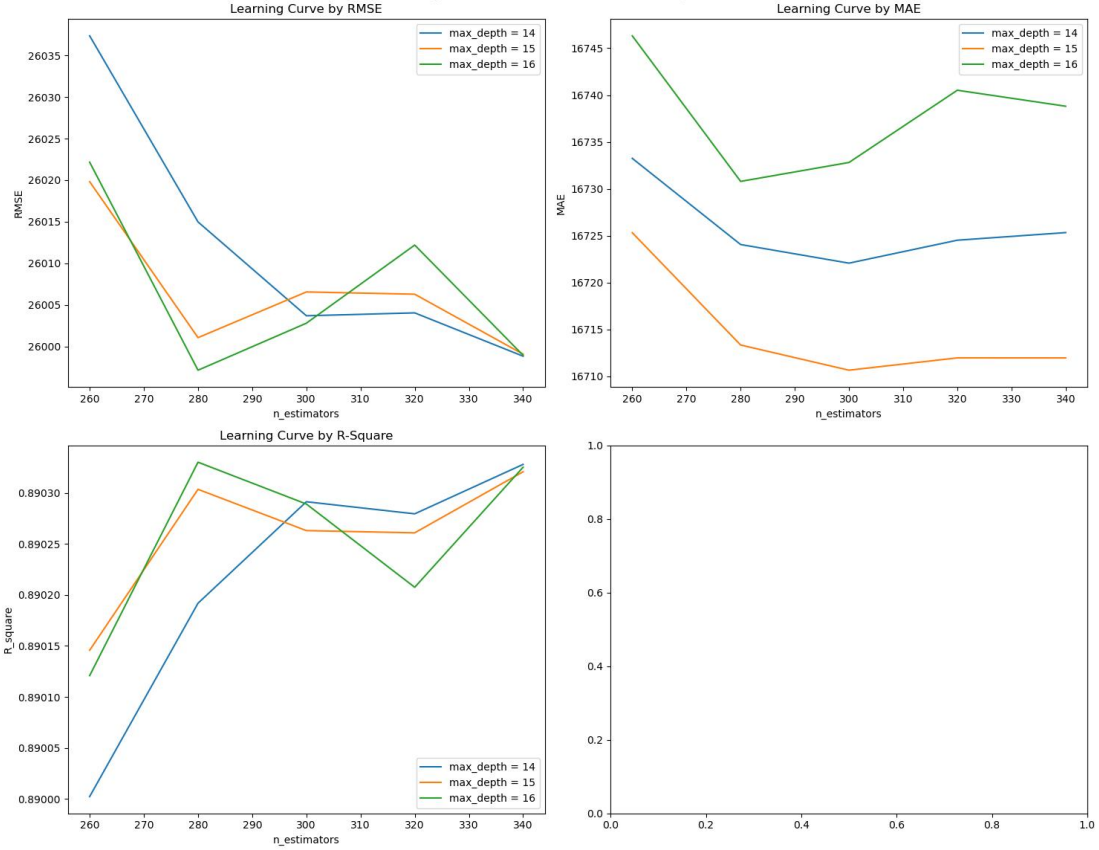


Table 9 Evaluation of Random Forest Regression Model

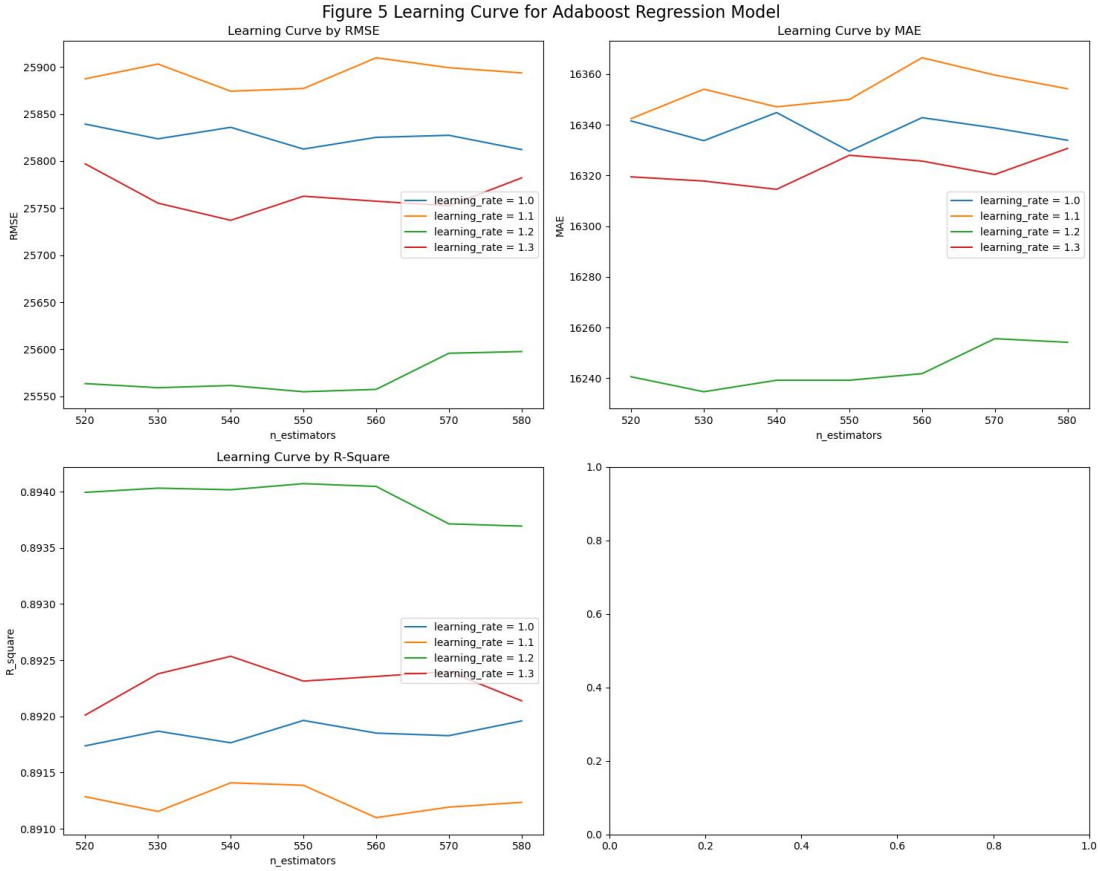
Model	Data Set	RMSE	MAE	R2
Random Forest Regression	Training	25578.56	16280.45	0.893
	Testing	31160.96	18268.03	0.861

4.1.3 Adaboost

Table 10 Refinement Step of Adaboost Regression Model

Order	Refinement Step	Details
1	Feature Engineering	● Label encoded categorical variables ● Target Variable is SalePrice
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.ensemble import AdaBoostRegressor
4	Hyper-parameters Adjusting	● 'n_estimators':[*range(100,601,50)] ● 'learning_rate':np.arange(0.1,1.1,0.1) ● 'estimator' = custom_decision_tree, the parameters of custom_decision_tree follow that of previous random forest regression model ● Fine tuning after Coarse tuning
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● RMSE ● MAE ● R2

Figure 5 Learning Curve for Adaboost Regression Model



In coarse tuning， {'n_estimators': 500, 'learning_rate': 1} were the optimal parameters. The Fine tuning was shown as Figure 5. The final parameter 'n_estimators' :530, 'learning_rate' :1.2.

Table 11 Evaluation of Adaboost Regression Model

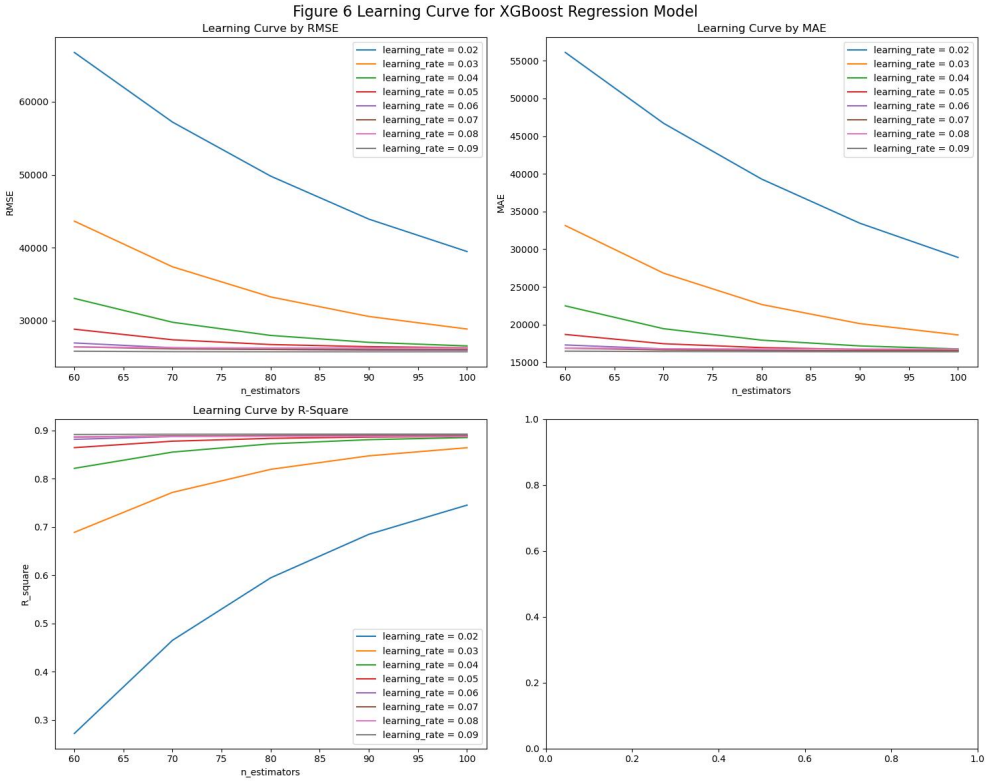
Model	Data Set	RMSE	MAE	R2
Adaboost Regression	Training	25909.73	16366.47	0.894
	Testing	30220.37	17871.40	0.872

4.1.4 XGBoost

Table 12 Refinement Step of XGBoost Regression Model

Order	Refinement Step	Details
1	Feature Engineering	● Label encoded categorical variables ● Target Variable is SalePrice
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from xgboost import XGBRegressor
4	Hyper-parameters Adjusting	● 'n_estimators':[*range(60,101,10)] ● 'learning_rate':np.arange(0.02,0.1,0.01) ● 'subsample':[0.93] ● 'max_depth':[16] ● 'booster':["gbtree"] ● XGBoost have many Hyper-parameters. To accelerate the parameter tuning process, only adjust a few relatively important parameters and set some empirical values.
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● RMSE ● MAE ● R2

Figure 6 Learning Curve for XGBoost Regression Model



The Fine tuning was shown as Figure 6. The final parameter {'learning_rate': 0.09, 'max_depth': 16, 'n_estimators': 100, 'subsample': 0.93}

Table 13 Evaluation of XGBoost Regression Model

Model	Data Set	RMSE	MAE	R2
Adaboost Regression	Training	25749.96	16384.85	0.892
	Testing	37022.39	18984.20	0.795

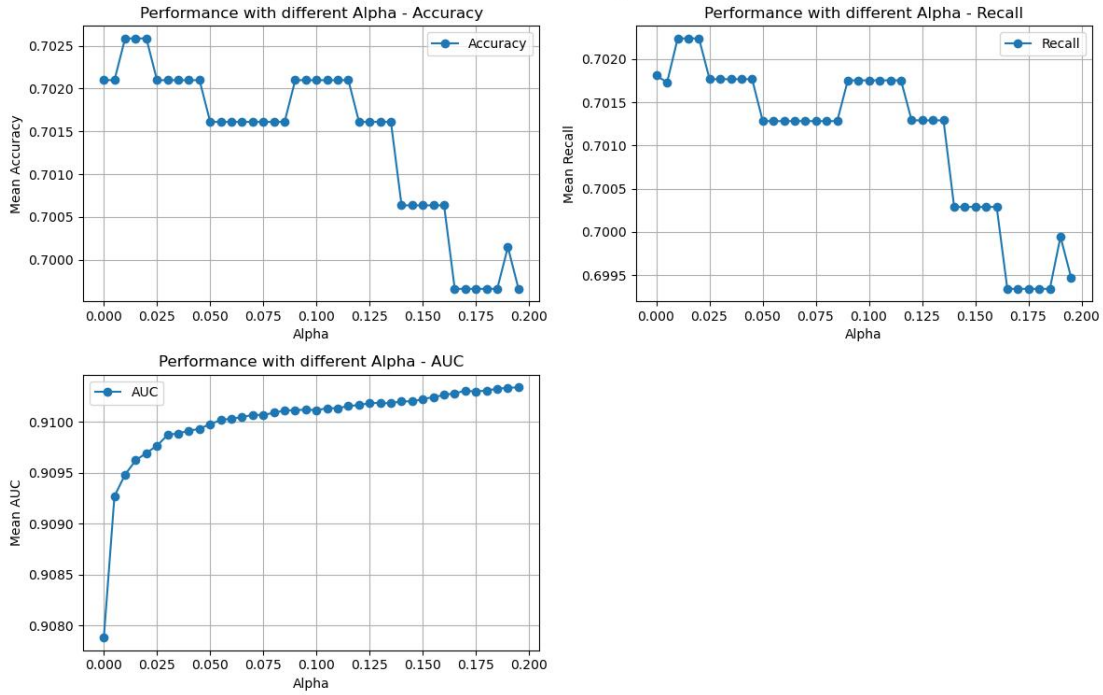
4.2 Classification Problem

4.2.1 Naive Bayes

Table 14 Refinement Step of Naive Bayes Classification Model

Order	Refinement Step	Details
1	Feature Engineering	● Continuous Variable Discretization ● Created dummy variables for categorical variables. ● Target Variable is PriceLevel
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.naive_bayes import MultinomialNB ● Input data are all discrete and Multinomial Naive Bayes is suitable for handling classification problems with multiple categorical features
4	Hyper-parameters Adjusting	● 'alpha':np.arange(0.001,0.2,0.002)
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	Considering that the sample categories are balanced, accuracy, recall, and AUC are chosen as performance evaluation metrics. ● Accuracy Accuracy measures the proportion of correctly predicted samples to the total number of samples. The higher the accuracy, the greater the level of matching between the model's predictions and the true results. ● Recall Recall measures the proportion of correctly predicted samples of a specific class to the total number of actual samples in that class. The higher the recall, the stronger the model's ability to cover positive samples, scoring = "recall_macro". ● AUC AUC plots the Receiver Operating Characteristic (ROC) curve based on the model's predicted recall (sensitivity) and false positive rate (1-specificity), then calculating the area under the curve. A higher AUC value indicates a stronger ability of the model to distinguish between positive and negative sample, scoring="roc_auc_ovr".

Figure 7 Learning Curve of Naive Bayes Classificaiton Model



The optimal hyper-parameter 'alpha': 0.0101

Table 15 Evaluation of Naive Bayes Classification Model

Model	Data Set	Accuracy	Recall	AUC
NaiveBayes Classification	Training	73.37%	73.21%	0.913
	Testing	70.26%	70.22%	0.909

4.2.2 SVM

Table 16 Refinement Step of SVM Classification Model

Order	Refinement Step	Details
1	Feature Engineering	● Standardized numeric variables ● Created dummy variables for categorical variables. ● Target Variable is PriceLevel
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.svm import SVC
4	Hyper-parameters Adjusting	● 'kernel':["linear","rbf"] ● 'gamma':np.logspace(-10, 1, 15) ● 'C':np.linspace(0.05,2,10) ● Fine tuning after Coarse tuning
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● Accuracy ● Recall ● AUC

In coarse tuning，{'C': 1.133, 'gamma': 0.0439, 'kernel': 'rbf'} were the optimal parameters. The Fine tuning was shown as Figure 8. The final parameter {'C': 1.373, 'gamma': 0.03, 'kernel': 'rbf'}.

Figure 8 Learning Curve for SVM Classification Model

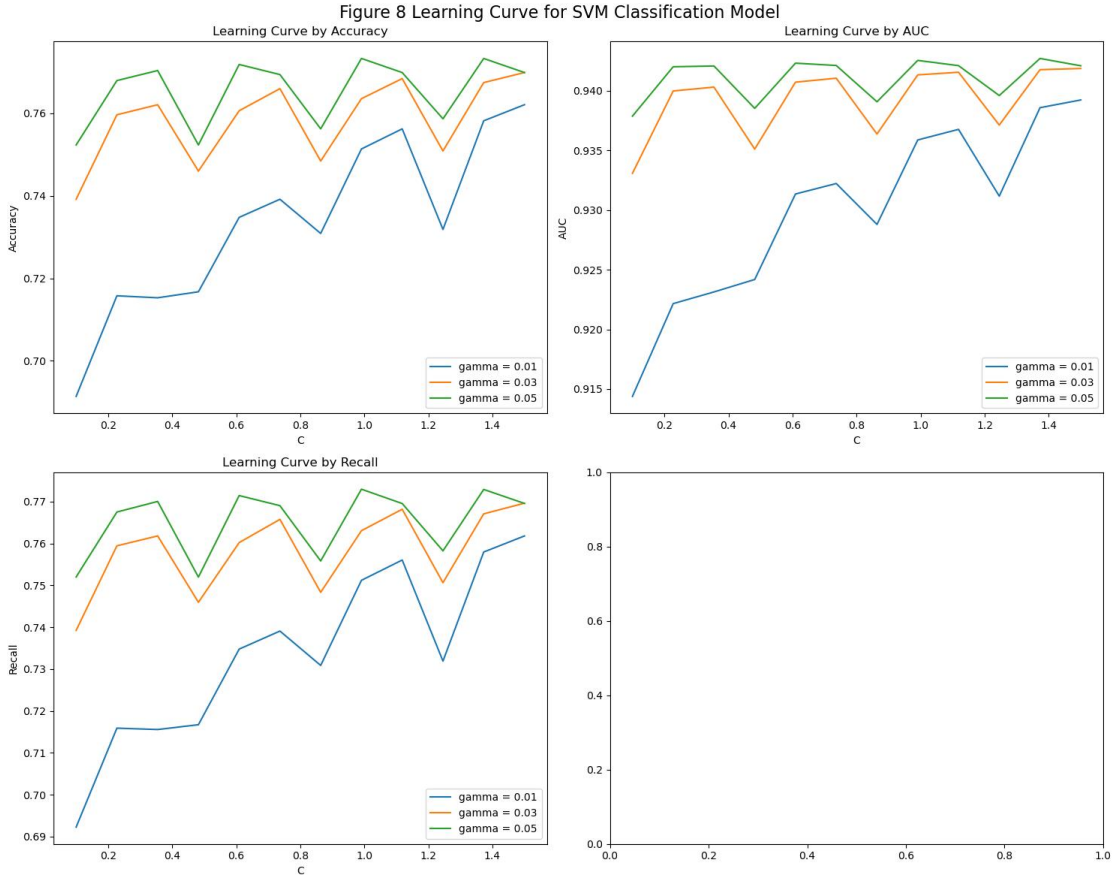


Table 17 Evaluation of SVM Classification Model

Model	Data Set	Accuracy	Recall	AUC
SVM Classification	Training	79.51%	79.40%	0.953
	Testing	77.93%	77.92%	0.952

4.2.3 Random Forest

Table 18 Refinement Step of Random Forest Classification Model

Order	Refinement Step	Details
1	Feature Engineering	● Label encoded categorical variables ● Target Variable is PriceLevel
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.ensemble import RandomForestClassifier
4	Hyper-parameters Adjusting	● 'n_estimators':[*range(71,802,10)] ● 'max_depth':[*range(13,18,1)] ● 'min_samples_leaf':[*range(1,3,1)] ● 'min_samples_split':[*range(1,6,1)] ● Fine tuning after Coarse tuning
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● Accuracy ● Recall ● AUC

In coarse tuning，{'n_estimators': 161} were the optimal parameters. The Fine tuning was shown as Figure 9. The final parameter {'max_depth': 14, 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 201}

Figure 9 Learning Curve for Random Forest Classification Model

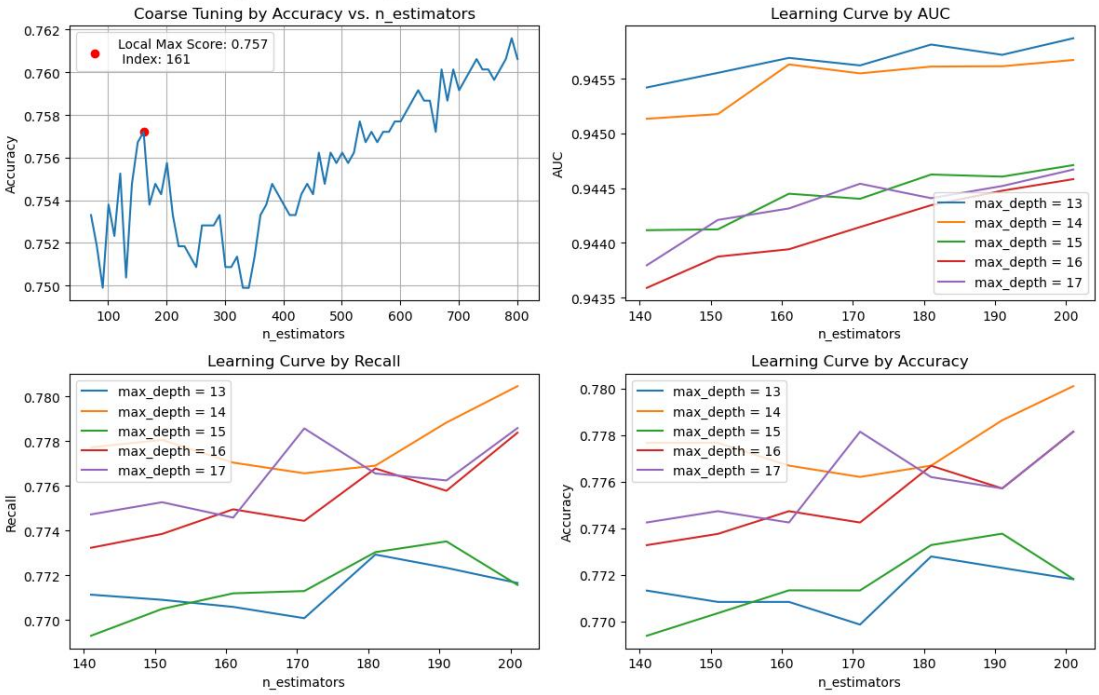


Table 19 Evaluation of Random Forest Classification Model

Model	Data Set	Accuracy	Recall	AUC
Random Forest Classification	Training	79.76%	79.61%	0.955
	Testing	78.95%	78.94%	0.949

4.2.4 Adaboost

Table 20 Refinement Step of Adaboost Classification Model

Order	Refinement Step	Details
1	Feature Engineering	● Label encoded categorical variables ● Target Variable is PriceLevel
2	Sample Spliting	● Train set 70% ● Test set 30% ● random_state = 0, which ensures that all model training and testing samples are consistent
3	Model Building	● from sklearn.ensemble import AdaBoostClassifier
4	Hyper-parameters Adjusting	● 'n_estimators':[*range(51,502,10)] ● 'Learning_rate':np.arange(0.1,3,0.2) ● Fine tuning after Coarse tuning
5	Cross-Validation	● 5 folds ● random_state = 0, which ensures that all model use the same data to cross validation
6	Evaluation	● Accuracy ● Recall ● AUC

In coarse tuning，{'n_estimators': 211} were the optimal parameters. The Fine tuning was shown as Figure 10. The final parameter is {'learning_rate':0.8, 'n_estimators': 231}.

Figure 10 Learning Curve for Adaboost Classification Model

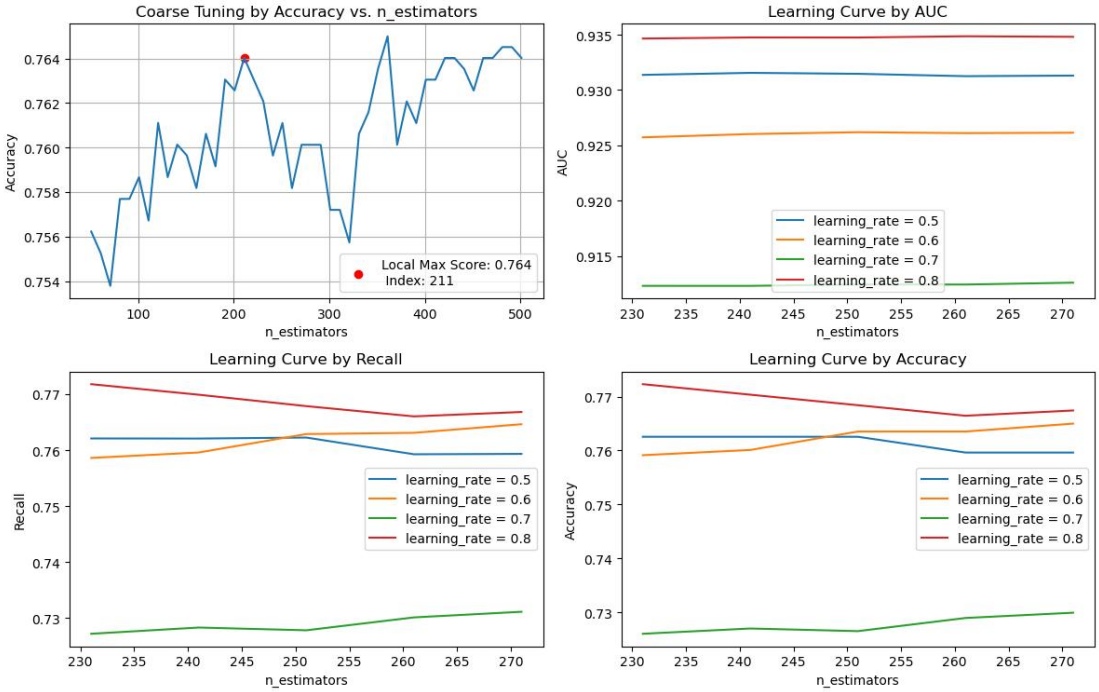


Table 21 Evaluation of Adaboost Classification Model

Model	Data Set	Accuracy	Recall	AUC
Adaboost Classification	Training	77.23%	77.18%	0.939
	Testing	76.45%	76.43%	0.935

5 Results Interpretation

5.1 Regression Model Interpretation

Table 22 Comparison of Regression Models

Model	Data Set	RMSE	MAE	R2
Linear Regression	Training	25915.02	16411.02	0.891
	Testing	35802.52	19490.85	0.815
Ridge Regression	Training	29357.84	17711.94	0.860
	Testing	35532.72	19330.66	0.818
Random Forest Regression	Training	25578.56	16280.45	0.893
	Testing	31160.96	18268.03	0.861
Adaboost Regression	Training	25909.73	16366.47	0.894
	Testing	30220.37	17871.40	0.872
XGBoost Regression	Training	25749.96	16384.85	0.892
	Testing	37022.39	18984.20	0.795

(1) Adaboost

In the test dataset, Adaboost Regression model achieves the smallest RMSE (30220.37), indicating a relatively small gap between the predicted house prices and the actual values. Additionally, it boasts the smallest MAE (17871.40), reflecting a lower average prediction error for house prices.The highest R2 (0.872) indicates the model's exceptional ability to explain the

variance in house prices and showcases its excellent fitting performance.

(2) Random Forest

Random Forest achieves similar performance metrics to Adaboost with only 280 DT (Decision Tree) estimators, while Adaboost uses 530 DT estimators. Moreover, Random Forest can be parallelized during the construction process because each decision tree is built independently. While AdaBoost is a sequential algorithm, where each base classifier's training depends on the results of the previous classifier. Therefore, Random Forest generally has higher computational efficiency compared to AdaBoost.

(3) Linear Regression

Although this model performs worse than all the other models, its results are acceptable. The Linear Regression model is simple and easy to understand. If high prediction accuracy is not a top priority, it can be a good choice.

(4) Ridge Regression

Ridge Regression's regularization can be helpful when dealing with multicollinearity. When Linear Regression and Ridge Regression have almost the same result, it may means preprocessing the data is effective and the regularization may not have a significant impact.

(5) XGboost Regression

As anticipated in 3.1, this model's performance is limited on a dataset with 2930 samples.

5.2 Classification Model Interpretation

Table 23 Comparison of Classification Models

Model	Data Set	Accuracy	Recall	AUC
Naive Bayes Classification	Training	73.37%	73.21%	0.913
	Testing	70.26%	70.22%	0.909
SVM Classification	Training	79.51%	79.40%	0.953
	Testing	77.93%	77.92%	0.952
Random Forest Classification	Training	79.76%	79.61%	0.955
	Testing	78.95%	78.94%	0.949
Adaboost Classification	Training	77.23%	77.18%	0.939
	Testing	76.45%	76.43%	0.935

(1) Random Forest

On the test dataset, the Random Forest model achieves the highest accuracy (78.95%) and the highest recall (78.94%), indicating its overall superior performance in classification. It also demonstrates strong ability in identifying positive samples correctly.

(2) SVM

SVM Classification model has the highest AUC value (0.952), indicating its capability to effectively distinguish between positive and negative samples. When SVM and Random Forest have similar performance metrics, the higher model complexity of SVM makes it less likely to be the best model choice.

(3) Naive Bayes

Its performance is the worst among all models. Although the Polynomial Naive Bayes classifier excels at handling discrete feature variables, when dealing with continuous data, it requires binning the continuous variables into discrete ones. This process can lead to information loss and reduce the model's classification accuracy.

(4) Adaboost

The result of Adaboost is similar with SVM and Random Forest. In the price range classification problem, some outliers may have limited the performance of Adaboost. Similar to the regression problem, Adaboost uses 231 DT estimators, while Random Forest only uses 201 decision tree estimators.

5.3 Conclusion

For the research purpose, this project leans towards selecting Adaboost Regression as the best model to predict house prices and Random Forest Classification as the best model to classify the price range.

Reference

De Cock, D. 2011. Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3).

Downey, A.B., Loukides, M.K., Blanchette, M., Kersey, A., Montgomery, K. and Demarest, R., 2015. *Think stats : exploratory data analysis.* Sebastopol, California: O'Reilly.

Ahsan, M. M., Mahmud, M. P., Saha, P. K., Gupta, K. D., & Siddique, Z., 2021. Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, 9(3), pp.52-59.

Galli, S., 2022. *Python feature engineering cookbook, Second edition.* Birmingham, England: Packt Publishing, Limited.

Schapire, R.E., 2013. *Explaining AdaBoost, Empirical Inference Festschrift in Honor of Vladimir N*. Vapnik, Springer Berlin Heidelberg , Berlin, Heidelberg.

Avanijaa, J., 2021. Prediction of House Price Using XGBoost Regression Algorithm. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(2), pp.2151-2155.