APPLIED STATISTICS AND MACHINE LEARNING

CA2 Assignment

Submitted by.

Rahul Pagar

Submitted on
13/12/2024

Table of Contents

Report:

Supervised Machine Learning- Regression

Objective

Analyse a dataset from a problem domain in depth, and select appropriate statistical models, tools, and techniques to derive insights regarding the dataset and domain. Demonstrate a critical understanding of the fundamentals of big data analytics. Effectively extract, transform, interrogate, and analyse large datasets.

Identify dataset for the assignment

For this assignement we used kc_housing dataset which is downloaded from Kaggle. This dataset is publicly available.

Importing Required Libraries

Import libraries essential for data handling and model building. Common libraries include:

pandas: For data manipulation and analysis.

sklearn. preprocessing: For data preprocessing.

sklearn. preprocessing import StandardScaler: for data scaling

linear model: for implementing linear regression models

Data Import

Import the dataset into the Python/Colab environment.

Our housing data file is in csv format, so we used **read_csv** to import file into python/colab environment.

We used **.head()** function to verify the imported file is in readable format. And to ensure the same file on which we will be operating. To see the rows output we can see the any number in parenthesis with head function.

Data Understanding

This step is performed to understand the dataset structure datatypes, statistical summary, rows and columns information.

I used **.shape** function to get the information about 21 columns and 21612 rows.

Then we used **.describe()** function is used to get the statistical summary of dataset.

The .**info()** can be used to understand the any null values in dataset also it gives information about column names and their data types, the number of non-null values.

Question 1. Data Preparation (What steps would you take to prepare your data)?

Answer:

Steps involved in data preparation are as follows:

1. **Data inspection and cleaning:**

- This step is performed to understand the dataset structure datatypes, statistical summary, rows and columns information.
- Cleaning step is performed to address the missing values in the dataset.
- No missing values in dataset as we get this information in data understanding step with the help of **.isnull().any()** function.

2. **Data encoding:**

- This step is performed to convert the categorical variables into numerical variables as we know machine learning algorithms can handle only numerical data.
- There are 20 numerical and 01 categorical variables in a dataset.
- Date column data type is object and this feature does not add any value for predicting the price so we drop this column.

3. **Construct the co-relation matrix and plot heat map:**

- This step uses a correlation matrix and a heatmap to analyse and show the relationships between different features in the dataset.
- Correlation matrix plotted to get the correlation between features/ variables.
- To plot the correlation matrix, we used. corr**()** function.
- A **heatmap** is generated after calculating the correlation matrix to provide a **visual representation** of the pairwise correlations between features.
- A heatmap allows you to quickly identify patterns and the strength of relationships between features. High correlations (either positive or negative) are easily spotted due to the colour gradient.
- For plotting heat map we used this function f=figure_factory.create_annotated_heatmap(cor.values,list(cor.columns),list(cor.columns),cor.round(2).values,showscale=True)
- if features are highly correlated (typically above 0.9 or below -0.9), the heatmap helps to visually flag such issues, which could lead to multicollinearity in linear regression.
- With heat map we identified **sqft_above** this feature has high correlation with another feature which might lead to multipolarity where both features provide the redundant information. So, we dropped **sqft_above** feature from our dataset.

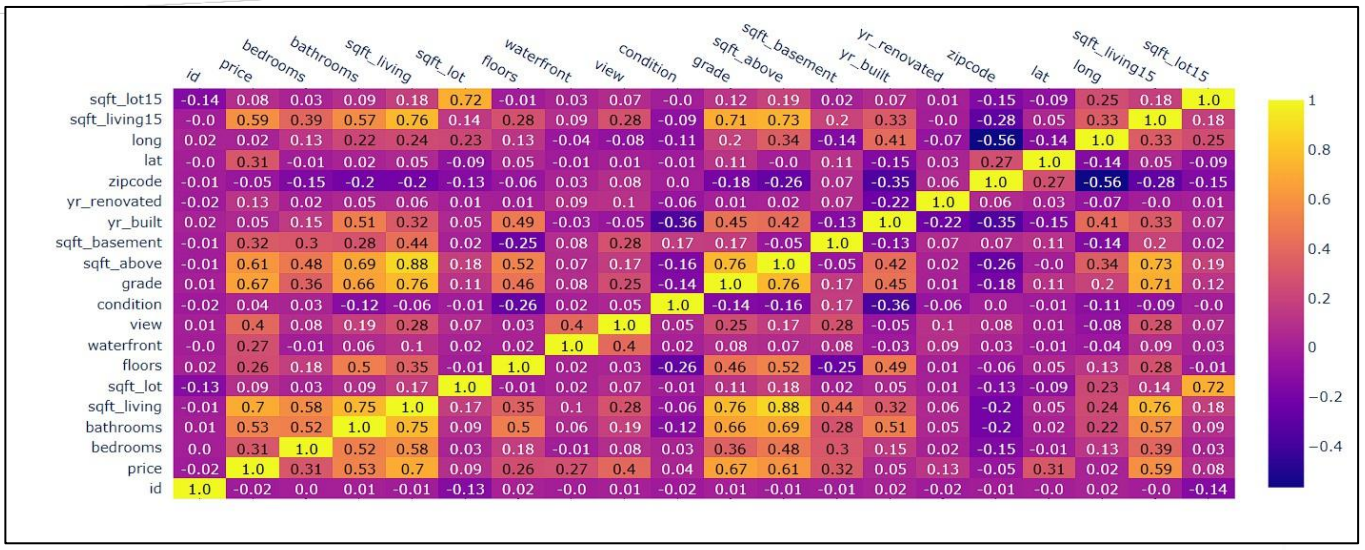

Fig1. Heatmap

4. Selecting dependent and independent variable:

- This step offers efficient modelling and data analysis. Making better decisions about the data and analyses, which results in more precise predictions and insights, is made easier by having a better understanding of dependent and independent variables.
- The dependent variable is those that the model is trying to predict.
- Independent variable is variable that is used to predict the dependent variable.
- In housing dataset price is dependent variable. It is assigned to “y” variable.
- In housing dataset except price column all remaining columns are independent variable. Excluding price column remaining columns assigned to “x” variable.
- New variables x and y created and these variables are independent and independent variables.

5. Data scaling / Standardisation:

- This step is performed to normalize the range of values in dataset. This is done by transforming the data into consistent scale. Scaling helps ensure that each feature contributes equally to the analysis process.
- Data scaling is performed on independent variable (x)
- The **StandardScaler().fit_transform()** is used to scaled the independent variable.

Question 2: Impact of L1, L2, and elastic net regularization on linear regression coefficients, performance, and interpretability. (Note - models built with fewer variables are considered more interpretable)

Answer:

Linear Regression is applied to housing dataset to predict the price.

The impact of L1, L2, and elastic net regularization on linear regression models can be significant in terms of coefficient estimation, model performance, and interpretability.

- Regularization techniques, such as lasso, ridge and elastic net, are used in linear regression to prevent overfitting and improve model performance.
- L1 regularization, also known as Lasso, sets some coefficients to zero, effectively selecting a subset of features, thereby increasing model interpretability.
- L2 regularization, or Ridge regression, shrinks all coefficients, reducing their magnitude.

- **Linear regression without penalty:**
  Initially linear regression is applied without penalty.
  The SGDRegressor was used to create the first linear regression model without any regularization features (penalty=None). Grid search underwent hyperparameter adjustment using the parameters max_iter (maximum permitted iterations) and eta0 (learning rate).

```
# Linear Regression
LR=linear_model.SGDRegressor(random_state=1,penalty=None) # Initialize the SGDRegressor (Stochastic Gradient Descent Regressor) for linear regr
# 'random_state=1' ensures reproducibility and 'penalty=None' means no regularization is applied to the model.
hyper_param={'eta0':[0.001,0.01,0.1,1],'max_iter':[1000,2000,3000,4000]} # # Define the hyperparameters to tune for grid search. 'eta0' is the
# Initialize GridSearchCV to perform hyperparameter tuning on the 'LR' model.
# It will search over the specified hyperparameters, using 5-fold cross-validation ('cv=5').
# The scoring metric is R^2, which measures the goodness of fit of the model (higher values are better).
grid_search=GridSearchCV(estimator=LR,param_grid=hyper_param,scoring='r2',cv=5)
grid_search.fit(x_st,y) # # Fit the model to the data (x_st is the feature matrix and y is the target vector).
# This will perform grid search and train the model on the best hyperparameters found.
```

After hyper parameter tunning we got eta0 as 0.001 and max_iter as 1000.

```
LR1=linear_model.SGDRegressor(random_state=1,penalty=None,eta0= 0.001, max_iter=1000) # Fitting the Linear model with boptimised parameters

LR1.fit(x_st,y) # Fitting the Linear model without penalty and with optimal parameters.
LR1.score(x_st,y) # R^2 score tells how much variation in the dependent variable is explained by independent variable.

0.6997560910818184
```

Model (LR1) fitted with optimal parameters (eta0=0.001, max_iter=10000. With these parameters we got **R^2 Score** as 0.6997. It means 69.97% variation in price is explained by independent variables.

- **Elastic net regularization:**
  Elastic net combines both L1 and L2, offering a balance between feature selection and shrinkage.
  Then linear regression is applied with penalty we used elastic net as it combines both lasso and ridge regression and following parameters tune to get the optimal parameters.

Hyperparameter tunning:
We used **SGDRegressor** as this type of linear regression model uses Stochastic Gradient Descent (SGD) to optimize the model. It's efficient for large datasets.
Then we use penalty as “elasticnet” as it combines both regularization technique.
We used **hyper_param** dictionary specifies a set of hyperparameters to test during the tuning process. eta0, max_iter, alpha, l1_ratio.
eta0 is the initial learning rate it determines the how large the steps are when updating the models’ coefficients. Max_iter explains is the maximum number of iterations. l1_ratio determines the mix between L1 and L2 regression.
We used GridSearchCV function to tune the eta0, max_iter, alpha, l1_ratio. (This function tries out each combination of the specified values and fits the model).
We used scoring = r2. This score measures how well the model fits the data.
This function also uses cross validation CV=5.By using five splits for cross-validation, the model is tested on various dataset segments to ensure consistent performance evaluation.

```
# Adding the penalty in the model
LR2=linear_model.SGDRegressor(random_state=1,penalty='elasticnet') # Linear regression without the regularisation
hyper_param={'eta0':[0.0001,0.001,0.01,0.1,1],'max_iter':[1500,2500,3000,3500],'alpha':[0.001,0.01,0.1,1],'l1_ratio':[0.20,0.30,0.40,0.50,0.60]}
grid_search1=GridSearchCV(estimator=LR2,param_grid=hyper_param,scoring='r2',cv=5) # R^2 tells how the good the linear model has fitted the data
grid_search1.fit(x_st,y) # Fitting the linear model
```

Linear regression with elastic net we got following parameters after tunning

Optimal Parameters after hyperparameter tunning	
alpha	0.01
eta0	0.001
l1_ratio	0.40
max_iter	1500

Model again fitted with these optimal parameters and we got $\mathbf{R^2}$ value as 0.6998. It means 69.98% variation in price is explained by independent variables.

```
LR3.fit(x_st,y)# Fitting the Linear model with penalty elasticnet and using the bestparameters from GridsearchCV
print(LR3.score(x_st,y)) # R^2 score tells how much variation in the dependent variable is explained by independent variable, i.e.69.96% of var
print('Beta_0:',best_model.intercept_)
#best_model.coef_
df=print(DataFrame(zip(x.columns,best_model.coef_),columns=['features','Beta coefficients']))

0.6997224992951159
Beta_0: [540085.75363621]
```

By observing the coefficients from the elastic net model, we can understand that:

The positive coefficients are bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, yr_renovated, lat, sqft_living15.These positive coefficients indicates a strong relationship with price, as these values increases price tend to increase.

The negative coefficients are id, bedrooms, sqft_basement, yr_built, zipcode, long, sqft_lot15 these features has negative impact of price.

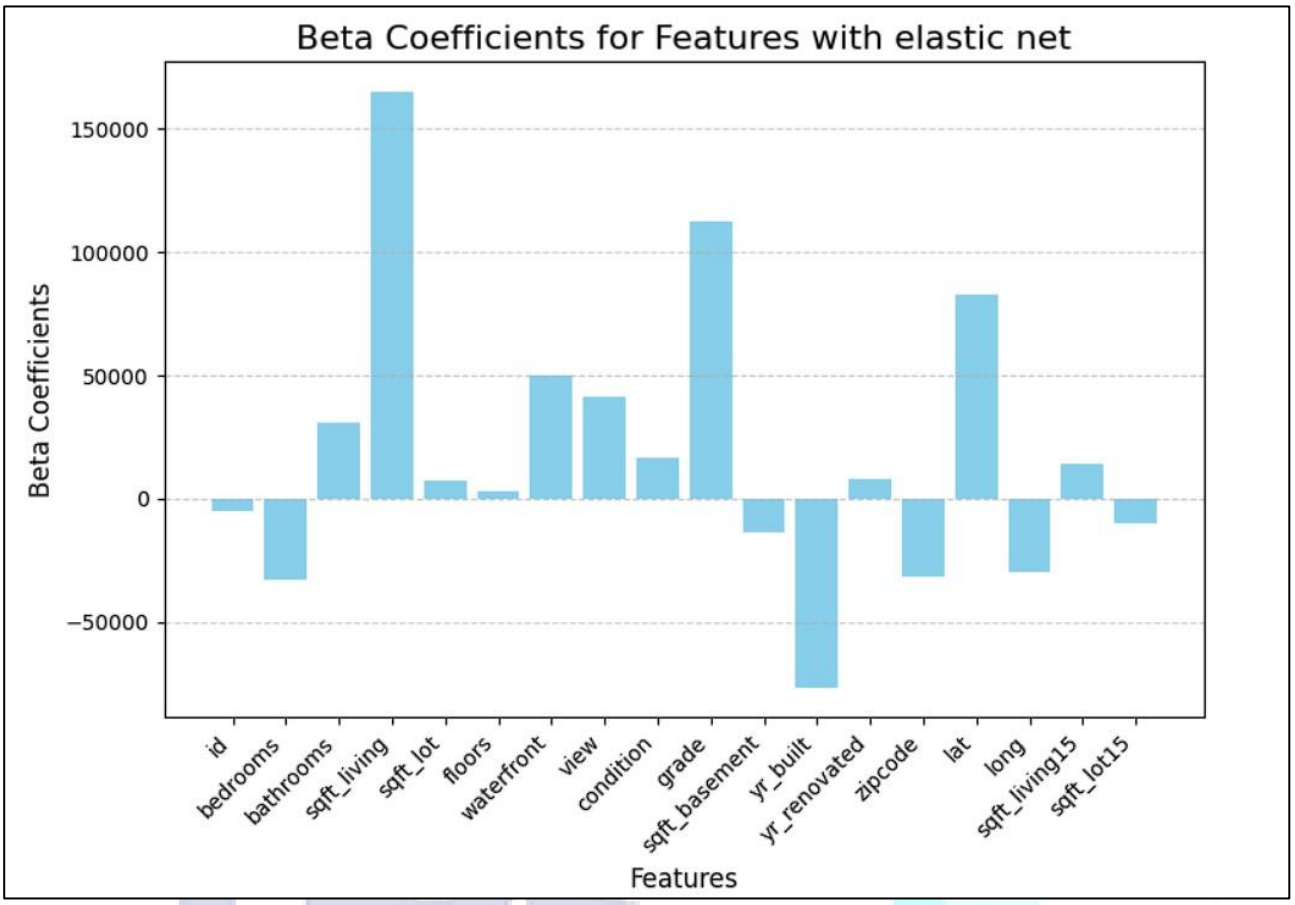

Fig2. Bar chart showing features and Beta coefficients

- **Ridge regression(L2):**
  L2 regularization adds a penalty proportional to the **square of the coefficients**. Unlike Lasso, Ridge does not shrink coefficients to zero. Instead, it **reduces the magnitude** of all coefficients more uniformly, leading to a more balanced solution.

  Hyperparameter tunning:
  A linear regression (LR_r) is initialised and We used **SGDRegressor** as this type of linear regression model uses Stochastic Gradient Descent (SGD) to optimize the model. It's efficient for large datasets.
  Then we use penalty as “elasticnet” as it combines both regularization technique.
  We used **hyper_param** dictionary specifies a set of hyperparameters to test during the tuning process. eta0, max_iter, alpha, l1_ratio.
  eta0 is the initial learning rate it determines the how large the steps are when updating the models’ coefficients. Max_iter explains is the maximum number of iterations. l1_ratio determines the mix between L1 and L2 regression.
  We used GridSearchCV function to tune the eta0, max_iter, alpha, l1_ratio. (This function tries out each combination of the specified values and fits the model).
  We used scoring = r2. This score measures how well the model fits the data.
  This function also uses cross validation CV=7.By using seven splits for cross-validation, the model is tested on various dataset segments to ensure consistent performance evaluation.

Optimal Parameters after hyperparameter tunning	
alpha	0.01
eta0	0.005
l1_ratio	0.0
max_iter	1000

Ridge performs nearly as well as Lasso but has a slightly lower R^2 (0.6993). This could suggest it does not eliminate irrelevant features as effectively as Lasso

- **Lasso regression(L1):**
  L1 applies a penalty that is proportionate to the coefficients' absolute values. effectively carries out feature selection by encouraging sparsity by setting some coefficients to precisely zero.

  Hyperparameter tunning:
  A linear regression (LR_l) is initialised and We used **SGDRegressor** as this type of linear regression model uses Stochastic Gradient Descent (SGD) to optimize the model. It's efficient for large datasets.
  Then we use penalty as “elasticnet” as it combines both regularization technique.
  We used **hyper_param** dictionary specifies a set of hyperparameters to test during the tuning process. eta0, max_iter, alpha, l1_ratio.
  eta0 is the initial learning rate it determines the how large the steps are when updating the models’ coefficients. Max_iter explains is the maximum number of iterations.
  We used GridSearchCV function to tune the eta0, max_iter, alpha, l1_ratio. (This function tries out each combination of the specified values and fits the model).
  We used scoring = r2. This score measures how well the model fits the data.
  This function also uses cross validation CV=7.By using seven splits for cross-validation, the model is tested on various dataset segments to ensure consistent performance evaluation.

Optimal Parameters after hyperparameter tunning	
alpha	0.001
eta0	0.003
l1_ratio	1
max_iter	1000

The R-squared value (0.6997) indicates that Lasso performs well and is slightly better than Ridge. This may suggest that some features in your dataset are irrelevant or less impactful, and Lasso's feature selection is helping simplify the model

Impact on coefficients:

- L1 regularization adds a penalty proportional to the absolute value of the coefficients.
- This can shrink some coefficients exactly to zero, effectively performing feature selection. This means Lasso tends to produce sparse models with fewer non-zero coefficients.
- L2 regularization adds a penalty proportional to the **square of the coefficients**.
- Unlike Lasso, Ridge does not shrink coefficients to zero. Instead, it **reduces the magnitude** of all coefficients more uniformly, leading to a more balanced solution.
- Elastic Net combines the strengths of both L1 and L2 regularizations. L1 contributes to sparsity (zeroing out irrelevant coefficients). L2 ensures balanced shrinkage and avoids over-penalizing correlated features (which can happen in pure Lasso).

Conclusion:

Elastic Net and Lasso regression models were identified as the most accurate based on their respective R^2 scores of 0.6997. Among these, Lasso regression is particularly well-suited for our situation, as it not only enhances interpretability but also enforces sparsity by setting some coefficients to zero, effectively highlighting key features and aiding feature selection. In contrast, Ridge regression retained all features, which could introduce noise and reduce interpretability, despite having a nearly identical R^2 value 0.69930.

Elastic Net, with its R^2 of 0.6997, is especially suitable for this dataset, offering a balanced approach that combines high sparsity with feature subset stability

Question 3: Impact of L2 regularization on support vector regression performance and interpretability.

Answer:

The C parameter in Support Vector Regression (SVR) controls the level of L2 regularization applied to the model. It plays a crucial role in balancing model complexity and performance.

1. **Support vector regressor without regularization parameter:**

- SVR () is a Support Vector Regressor, which is a machine learning model used for regression task. CV=10.Cross-validation divides the dataset into 10 parts (folds). The model is trained on 9 parts and tested on the remaining part. This process is repeated 10 times, and the average score is calculated.
- Hyperparameters used for tunning kernel and epsilon.
- This also uses **GridSearchCV,** this automates the process of trying different combinations of hyperparameters to find the best ones.
- We used scoring = r2. This score measures how well the model fits the data.

```
from sklearn.svm import SVR
SVRegressor1 = SVR()
cv=10
Hparameters1= {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'epsilon': [100,1000,10000]}
grid_search2 = GridSearchCV(estimator=SVRegressor1, param_grid=Hparameters1, scoring='r2')
grid_search2.fit(x_st, y)

best_parameters = grid_search2.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search2.best_score_
print("Best result: ", best_result)
```

- We applied Support vector regression without regularization parameter. We fitted SVR with optimal parameters (kernel =linear and epsilon = 100) and we got R-squared value as 0.11 which is too low.
- This indicates that model is too simple and ignored the important patterns in the data. It also indicates that model was over regularized and excessively simplify the relationship between features and target variables.
- No regularization parameter (C) indicates SVR margin became too wide ignoring smaller deviation in the data. This leads to poor fitting. Without regularization parameter model became easy to interpret but not capturing relationship effectively. With this r square 0.11 model is underfitting.

2. Support vector regressor with regularization parameter:

- In method 2 we used support vector regressor with regularization parameter.
- We used SVRRegressor3 with CV=10. Cross-validation divides the dataset into 10 parts (folds). The model is trained on 9 parts and tested on the remaining part. This process is repeated 10 times, and the average score is calculated.
- We hyper tunned following parameters kernel, regularization parameter and epsilon. The epsilon specifies margin of tolerance around the predicted values.
-

```
from sklearn.svm import SVR
SVRegressor3= SVR()
cv=10
Hparameters2= {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [100,1000,10000], 'epsilon': [100,1000,10000]}
grid_search3 = GridSearchCV(estimator=SVRegressor3, param_grid=Hparameters2, scoring='r2')
grid_search3.fit(x_st, y)

best_parameters = grid_search3.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search3.best_score_
print("Best result: ", best_result)
```

- After tunning we got following optimal parameters:

Optimal Parameters after hyperparameter tunning	
Kernel	Poly
C	10000
epsilon	1000

- We fitted this SVR and we got following r squared value: 0.76 (76.10%) it means 76.10% variance in the price (target variable) is explained by the independent variables. With optimised regularization parameter the r squared value improved to 0.7610.
- With regularization parameter model became more flexible and captured the patterns in the dataset. Optimal regularization parameter giving balance bias and variance.
- Optimised regularised parameter retains the coefficients contributing to predict the price while it avoided underfitting as well prevented overfitting.
- With optimal regularization parameter SVR performance is improved with r squared value as 0.76. With this we can say that model focuses more on fitting the data accurately. This allows model to better reflect the actual relationship between features and the target variable.

Conclusion:

Initially, without a regularization parameter, the **Support Vector Regression (SVR)** model had a very low r squared value of **0.11**, meaning it wasn't capturing important patterns in the data and was **underfitting**.

After tuning the model's parameters and adding regularization parameter, we improved the r squared value to **0.76**, meaning the model now explains **76%** of the target variable's variance.

The optimized regularization made the model more flexible, balancing between underfitting and overfitting. This improvement allowed the model to capture the true relationship between the features and the target more accurately.

Question 4: If you were to implement random forest regression, then its comparative performance and interpretability with respect to regularized linear regression and regularized support vector regression models.

Answer:

- **Random forest regression is applied for predicting price.**

A random forest regression (RF_Regressor2) is initialised and We used RandomForestRegressor.
criterion='squared_error' The model uses squared error to measure the quality of each split in the decision trees.
max_features='sqrt': Each tree considers the square root of the total number of features when deciding how to split the data.
We used no_trees dictionary specifies a set of hyperparameters to test during the tuning process. n_estimators and max_features.
We used GridSearchCV is a tool that searches through all combinations of hyperparameters in param_grid to find the best-performing one.
scoring='r2': The model will be evaluated using the R-squared score, which measures how well the predictions match the actual values
CV=5 The dataset is split into 5 parts, and the model is trained and validated on different combinations.

```
from sklearn.ensemble import RandomForestRegressor
RF_Regressor1 = RandomForestRegressor(criterion='squared_error', max_features='sqrt', random_state=1)
no_Trees = {'n_estimators': [5,10,15,20,22,24], 'max_features':['sqrt','log2', None]}
grid_search4 = GridSearchCV(estimator=RF_Regressor1, param_grid=no_Trees, scoring='r2', cv=5)
grid_search4.fit(x_st, y)

best_parameters = grid_search4.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search4.best_score_
print("best_score: ", best_result)
```

After tunning we got following optimal parameters:

Optimal Parameters after hyperparameter tunning	
n_estimators	24
max_features	None

We fitted this random forest and we got following r squared value: 0.9798 (97.98%) it means 97.98% variance in the price (target variable) is explained by the independent variables. But this R-squared value indicates my model is overfitting.

To avoid overfitting, we applied important features selection:
In this method we select only most relevant features, Random Forest can become less complex with feature selection which helps to reduce the overfitting.

In feature selection we select the relevant features and remove the redundant and irrelevant features which helps the model to focus on important data and on meaningful patterns

```
from sklearn.ensemble import RandomForestRegressor
RF_Regressor2 = RandomForestRegressor(criterion='squared_error', max_features='sqrt', random_state=1)
no_Trees = {'n_estimators': [2,4,6,8,10,12,14], 'max_features':['sqrt','log2', None]}
grid_search5 = GridSearchCV(estimator=RF_Regressor2, param_grid=no_Trees, scoring='r2', cv=5)
grid_search5.fit(x_st, y)

best_parameters = grid_search5.best_params_
print("Best parameters: ", best_parameters)
best_result = grid_search5.best_score_
print("best_score: ", best_result)
Important_feature = Series(grid_search5.best_estimator_.feature_importances_, index=list(x)).sort_values(ascending=False) # Getting feature importances list for the best model
print(Important_feature)

Best parameters:  {'max_features': None, 'n_estimators': 14}
best_score:  0.8683068489265814
grade            0.356690
sqft_living      0.240884
lat              0.159495
long             0.070393
waterfront       0.029953
sqft_living15    0.029486
yr_built         0.028544
zipcode          0.015222
sqft_lot15       0.014145
sqft_lot         0.013511
view             0.009189
id               0.008678
bathrooms        0.007990
sqft_basement    0.005992
bedrooms         0.003317
condition        0.002835
floors           0.002029
```

We got following important features:

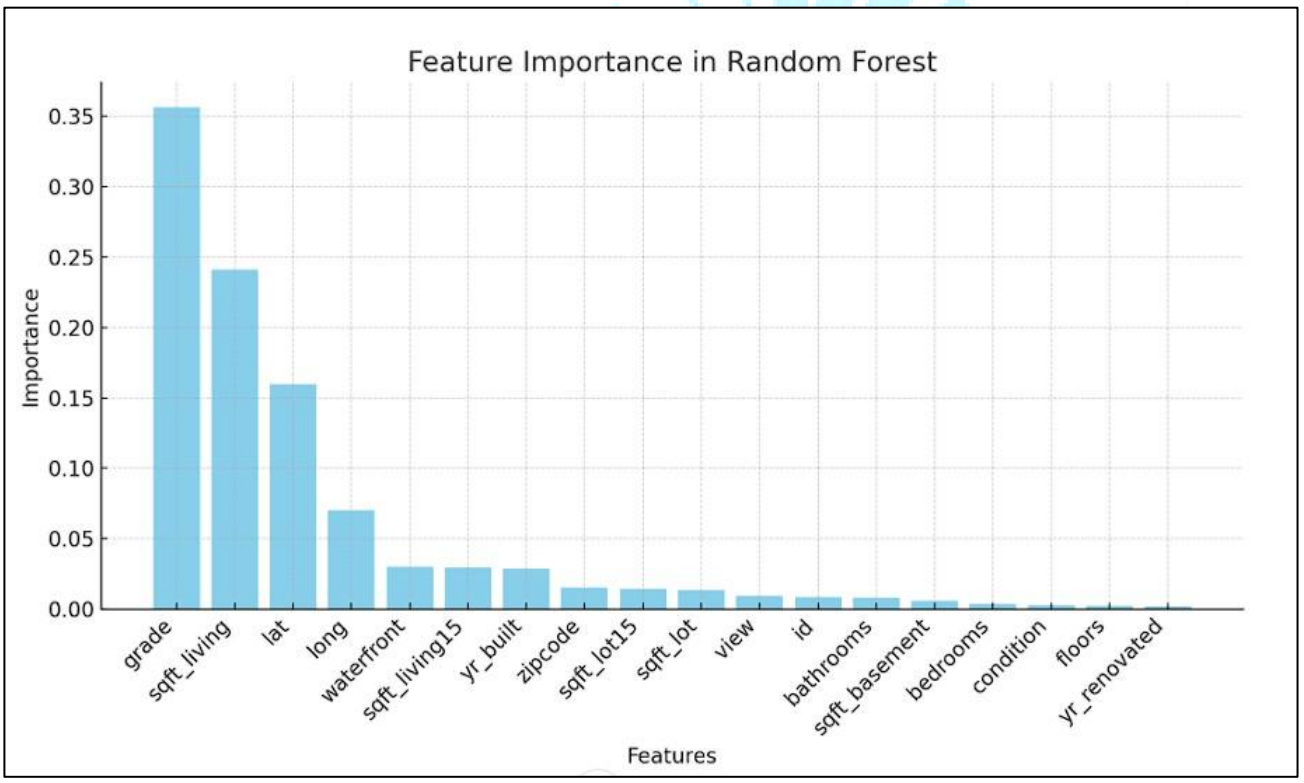

Fig3. Bar chart showing important features in random forest

Based on this data we selected following features 'grade', 'sqft_living', 'lat', 'long','waterfront' and created new variable(z). we transformed this by using StandardScaler().fit_transform(). With these features again hyperparameter tunning done. After tunning we got following optimal parameters: n_estimators=12 and mex_features= None.
We fitted random forest with important features and hyper tunned parameters and we got following r squared value: 0.9761 (97.61%).

Even after feature selection random forest is overfitting.

So, we are suggesting plotting heat map to reduce the overfitting as the heatmap helps to visually flag such issues, which could lead to multicollinearity in linear regression. Also, we can reduce the overfitting by principal component analysis.

- **Random forest regression vs Regularized regression:**
  With random forest regression R-squared value we got 0.9761. This indicates it explained the 97.61% of the variability in the data. Which is better than regularised linear regression 69%. Random forest regressor makes no assumptions for relationship between feature and target. It splits the data into small decision trees each making its own prediction then it averages these predictions for better result. But linear regression gives coefficients but random forest regressor doesn't provide coefficient like linear regression but it provides feature information. Which feature contribute the most during prediction. Each feature has a clear coefficient that shows its direct impact on the target. We can easily understand how changes in each feature influence the outcome. Higher R-squared value in random forest indicate that data has more non-linear relationship and interactions between features that linear regression cannot identify random forest can. Random forests are flexible and can capture these non-linear patterns because they make splits in the data. Random forest regressor is ensemble method due to which overall model structure become more complex than linear regression. Random forest is suitable for complex and non-linear data. Linear regression misses complex trends in the data while random forest regressor captures the complex patterns in the data. But if we compare the performance of random forest regressor over regularized linear regression R-squared value is better but model is overfitting.

- **Random forest regression vs Regularized support vector regression:**
  Random forest explains the 97.11% of the variability while SVR explains 76.10%. A higher r squared value suggest model males more accurate predictions. This gap means random forest regression explains the more variability in the target than regularized linear regression. When data has non-linear relationship and have complex patterns that a straight line cannot capture.
  This explains random forest captures more patterns in our data compared to regularised SVR and complexities. While SVR and random forest handles non linear relationship in the data SVR uses kernel function to handle the non-linear relationship. But SVR can struggle handling large complex data on the other hand random forest handles non-linear relationship and handle complex data. SVR works by minimizing error within a defined margin which can make it sensitive to noisy data. Random forests are less sensitive to noise because they use ensemble of trees and noisy data have less influence on the average prediction. SVR doesn't provide the clear insights into feature importance because model is built around support vector. Random forests can rank features by their importance to predictions, giving you better insights into which variables matter most.

- **Conclusion:**
  Based on the R-squared value of 97.10%, the Random Forest Regressor outperforms both Regularized Linear Regression (RLR) and Regularized Support Vector Regression (SVR). However, the Random Forest model is overfitting, meaning it captures not only general patterns but also noise in the data, leading to poor performance on unseen data. To address this, methods like Heatmap and PCA are recommended.
  Regularized SVR explains 76.10% of the variability, offering a balanced performance between underfitting and overfitting. However, it is less interpretable compared to RLR and Random Forest, as it focuses on finding a hyperplane without providing clear feature-target relationships. Identifying which features impact predictions is more challenging with SVR. In contrast, Random Forest highlights feature importance effectively.
  Regularized linear regression explains 69.97% of the variations. Also, it is more interpretable as in regularised linear regression, each feature has a clear coefficient that shows its direct impact on the target. We can easily understand how changes in each feature influence the outcome.
  Random Forest is Best for complex, non-linear data with high accuracy but with our data it is overfitting. So, we recommend methods like Heatmap and PCA to reduce the overfitting.
  Regularized SVR: Balanced performance for non-linear data but less interpretable. As, it does not provide a clear relationship between each feature and target variable
  RLR is most interpretable, with clear coefficients showing each feature's direct impact on the target.

Question 5: Performing a prediction for one of your models using new data.

Answer:

In our data “price” is a target variable (dependent variable) is a directly proportional to the positive coefficients. The positive coefficients bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, yr_renovated, lat, sqft_living15.These positive coefficients indicates a strong relationship with price, as these values increases price tend to increase.

Elastic Net: Beta_0 = 541004.82
Combining Lasso (L1) and Ridge (L2) regularization, Elastic Net strikes a balance between feature selection and weight reduction. The intercept is marginally lower than Ridge and Lasso's, indicating that the combination of the L1 and L2 penalties has produced a somewhat less flexible fit that balances generalization and complexity

Lasso regression: Beta_0 = 541004.81983264
Lasso regression emphasizes sparsity by shrinking some coefficients to zero. The slightly higher intercept might indicate that Lasso is compensating for the exclusion of certain predictors by slightly elevating the baseline (intercept).
Ridge regression: Beta_0 = 541599.12368911

Ridge regression penalizes large coefficients (L2) but doesn't drive them to zero. Its intercept is closer to Lasso, reflecting its tendency to maintain all predictors in the model but with smaller magnitudes.

The variations in the intercept values are generally small, suggesting that the regularization techniques are not drastically changing the baseline prediction. The differences might stem from how each regularization method optimally balances bias and variance.

Linear regression equation

Sr No	Model	Equation
1	ElasticNet	y = 540085.75363621+ (-3733.157225 x id) + (-31399.227453 x bedrooms) + (33239.361396x bathrooms) + (157002.776176 x sqft_living) + (5359.486021x sqft_lot) + (4995.345296 x floors) + (50406.853813 x waterfront) + (40312.246862 x view) + (17076.205967 x condition) + (113949.613418 x grade) + (-10818.017190 x sqft_basement) + (-76889.822714 x yr_built) + (8096.197710 x yr_renovated) + (-30268.301208 x zipcode) + (82646.610337 x lat) + (-29181.097284 x long) + (18592.505313 x sqft_living15) + (-10455.253370 x sqft_lot15)
2	Ridge Regression	y = 541599.12368911 + (-5269.940326 x id) + (-29930.705481 x bedrooms) + (31382.272570 x bathrooms) + (156432.007861 x sqft_living) + (8760.905234 x sqft_lot) + (3617.526649 x floors) + (49769.842025 x waterfront) + (41158.500491 x view) + (16638.996780 x condition) + (111729.699071 x grade) + (-12175.771762 x sqft_basement) + (-74241.038075 x yr_built) + (7842.432336 x yr_renovated) + (-31238.752134 x zipcode) + (81733.416184 x lat) + (-29142.797673 x long) + (16503.971523 x sqft_living15) + (-10342.662456 x sqft_lot15)
3	Lasso Regression	y = 541004.81983264 + (-4829.757228 x id) + (-32854.746442 x bedrooms) + (31106.163748 x bathrooms) + (164994.847510 x sqft_living) + (7538.948099 x sqft_lot) + (2937.654931 x floors) + (50304.094599 x waterfront) + (41169.247669 x view) + (16848.626448 x condition) + (112403.799110 x grade) + (-13709.332129 x sqft_basement) + (-76366.426857 x yr_built) + (7795.859177 x yr_renovated) + (-31659.339420 x zipcode) + (82877.677404 x lat) + (-29951.303389 x long) + (13920.283970 x sqft_living15) + (-10145.816870 x sqft_lot15)

Price prediction using regularized support vector regression:

In our case random forest regressor has a highest R-squared value but random forest is overfitting so we are not considering random forest for prediction. Regularized support vector regressor has a best R-squared value which is 76.10%. This model we have used for prediction.

New data considered for prediction.

```
# Prediction with New Data
import pandas as pd
new_data = pd.DataFrame({
    'id': [7129300524],
    'bedrooms': [5],
    'bathrooms': [3.5],
    'sqft_living': [1680],
    'sqft_lot':[9234],
    'floors': [2.1],
    'waterfront': [0],
    'view': [2],
    'condition':[5],
    'grade': [13],
    'sqft_basement':[1870],
    'yr_built':[2001],
    'yr_renovated':[2018],
    'zipcode':[98742],
    'lat':[47.9234],
    'long':[-122.343],
    'sqft_living15':[1721],
    'sqft_lot15':[4394]
})
```

With above value we used predict function with svr.regressor we predicted price.

```
# Prediction
prediction = svr_regressor.predict(new_data_scaled)  # Assuming RF_regressor_feature is the intended model
print("Prediction for New Data With Regularized Support Vector Regressor:", prediction)
```

```
Prediction for New Data With Regularized Support Vector Regressor: [461398.43513002]
```

Price prediction value we got with regularized SVR: 461398.43513002

After regularized support vector regressor elastic net and lasso regression is best based on R-squared value.

Price prediction with elastic net regression:

Price predicted considering same value as we considered while predicting with regularized SVR.

```
predict1=LR3.predict(new_data_scaled)
print("Prediction for New Data With Elastic net:", predict1)
```

```
Prediction for New Data With Elastic net: [540085.75363621]
```

Price prediction value we got with elastic net: 540085.75363621

Price prediction with lasso regression:

Price predicted considering same value as we considered while predicting with regularized SVR.

```
predict2=LR_lasso.predict(new_data_scaled)
print("Prediction for New Data With Lasso:", predict2)
```

```
Prediction for New Data With Lasso: [541004.81983264]
```

Price prediction value we got with Lasso regression: 541004.81983264

Conclusion:

Even though SVR is high R-squared value that elastic net and lasso regression but predicted value is significantly low.

Elastic Net (540,085.75): Elastic Net predicts a moderate price, combining L1 (Lasso) and L2 (Ridge) regularization to balance feature selection and coefficient shrinkage. It likely captures a compromise between bias and variance.

Lasso (541004.81983264): Lasso predicts a slightly higher price compared to Elastic Net. This could be due to its focus on sparsity, where certain predictors are eliminated entirely, leaving fewer but potentially stronger predictors in the model.

The small difference between Elastic Net and Lasso predictions suggests that the price prediction is relatively stable in the context of linear relationships.

Regularized SVR (461398.43513002): Regularized SVR predicts a significantly lower price. SVR is a nonlinear model, and its regularization allows it to find complex patterns. This might suggest that SVR is capturing a different underlying relationship. The large deviation in SVR's prediction indicates that SVR might be capturing patterns missed by the linear models.

Individual contribution

Name	Contribution
Rahul Pagar	Linear regression with elastic net and support vector regressor with and without regularization parameter
Ravi	Data identification from Kaggle and data preparation and price prediction with new data
Sanford Rodrigues	Linear regression without penalty and lasso and ridge regression
Satyanarayan Purohit	Random forest regressor for price prediction with optimal parameters and random forest with importance feature

Reference list

https://www.kaggle.com/datasets?search=kc_

https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.SGDRegressor.html