HOUSE PRICE PREDICTION

Mahjabin Hassan
ITCS 5156

Problem & Challenges

Housing prices depend on a myriad of factors like location, size, population, city etc. The purpose of the project is to find the best machine learning model that can predict house price .

Motivation

Housing price is an important factor that reflects the economy. There are so many ways housing prices can be explored and predicted using machine learning.

As someone who is new to machine learning, the topic seemed very interesting.

For the purpose of the project, I am using random forest regression model and comparing it with other regression models like Linear, Ridge and Lasso Regression.

Existing Related Approaches

There are different papers available on the topic using different techniques.

Regression, inference, neural networks, and deep learning are some of the most popular machine learning methods.

Method

The dataset for the project was obtained from Github.

https://github.com/pplonski/datasets-for-start/blob/master/house_prices/data.csv

Steps:

1. Load the dataset
2. Apply preprocessing/ data cleaning
3. Apply the different ML algorithms (Linear, Ridge, Lasso, Random Forest Regression)
4. Compare the results
5. Visualize the predictions of the best model.

Result & Observation

After loading the dataset, it can be seen that there are 1460 rows (data) and 81 columns (features).

Before data cleaning, data exploration was performed to answer the following questions:

Do we need all of the features?

Are there missing values in the dataset?

Next the repetitive features and the features which with the most missing values were removed.

Result & Observation(continued)

```
#checking for any missing values
check_NaN = df.isnull().values.any() # check for NaN values in dataframe
check_NaN
```

```
]:  True
```

```
#counting the total number of missing values for each feature in df
total = df.isnull().sum().sort_values(ascending=False)
total.head(20)
```

```
]:  PoolQC          1453
    MiscFeature     1406
    Alley           1369
    Fence           1179
    FireplaceQu      690
    LotFrontage      259
    GarageYrBlt       81
    GarageCond        81
    GarageType        81
    GarageFinish      81
    GarageQual        81
    BsmtFinType2      38
    BsmtExposure      38
    BsmtQual          37
    BsmtCond          37
    BsmtFinType1      37
    MasVnrArea         8
    MasVnrType         8
    Electrical         1
    Id                 0
    dtype: int64
```

Result & Observation(continued)

```
# deleting unnecessary/repetitive features from df and store the new df in df_new
df_new = df.drop(['PoolQC','MiscFeature', 'Alley', 'Fence','FireplaceQu','LotFrontage', 'Id',
                  'GarageYrBlt','GarageCond','GarageType', 'GarageFinish', 'GarageQual',
                  'BsmtFinType2', 'BsmtExposure','BsmtQual', 'BsmtCond', 'BsmtFinType1', 'MasVnrArea', 'MasVnrType'], axis=1)
df_new.head(5)
```

	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	...	EnclosedPorch	3SsnPorch	Screer
0	60	RL	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	...	0	0	
1	20	RL	9600	Pave	Reg	Lvl	AllPub	FR2	Gtl	Veenker	...	0	0	
2	60	RL	11250	Pave	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	...	0	0	
3	70	RL	9550	Pave	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	...	272	0	
4	60	RL	14260	Pave	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	...	0	0	

5 rows × 62 columns

Now only 62 features remained.

Result & Observation(continued)

```python
#keeping only numeric data for regression analysis
#df_house_n is the new datframe that will be used for Machine learning models

df_house_n = df_house.select_dtypes(include=['int64'])
df_house_n
```

	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	TotalBsmtSF	...	WoodDeckSF
0	60	8450	7	5	2003	2003	706	0	150	856	...	0
1	20	9600	6	8	1976	1976	978	0	284	1262	...	298
2	60	11250	7	5	2001	2002	486	0	434	920	...	0
3	70	9550	7	5	1915	1970	216	0	540	756	...	0
4	60	14260	8	5	2000	2000	655	0	490	1145	...	192
5	50	14115	5	5	1993	1995	732	0	64	796	...	40
6	20	10084	8	5	2004	2005	1369	0	317	1686	...	255

Next, only keeping the numeric values for our model, since I will be performing regression and random forest regression techniques.

Result & Observation(continued)

```python
# Splitting data into test and train data
# 80% train data and 20% test data

from sklearn.model_selection import train_test_split

def data_splitting(data, target):
    X_train, X_test, t_train, t_test = train_test_split(data,target, test_size=0.2, random_state=0)
    return X_train, X_test, t_train, t_test

X_train, X_test, t_train, t_test = data_splitting(
    data = df_house_n.iloc[:, :33],
    target = df_house_n['SalePrice']
)

print("Train data shape: {}".format(X_train.shape))
print("Train target shape: {}".format(t_train.shape))
print("Test data shape: {}".format(X_test.shape))
print("Test target shape: {}".format(t_test.shape))
```

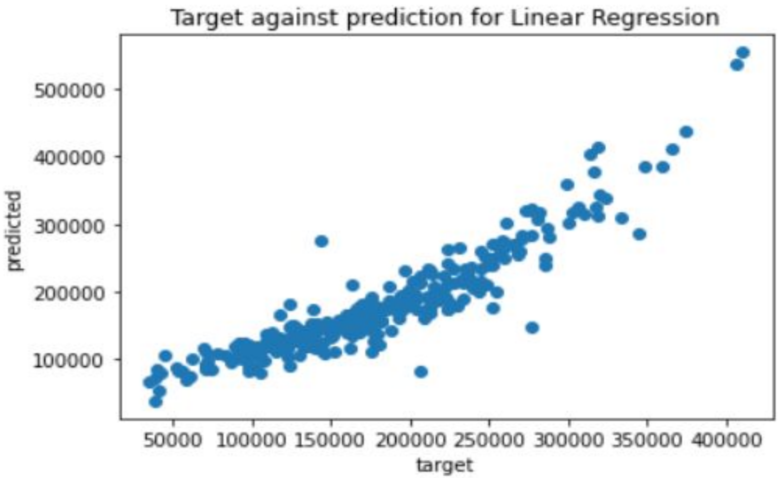
```
Train data shape: (1167, 33)
Train target shape: (1167,)
Test data shape: (292, 33)
Test target shape: (292,)
```

Result & Observation(continued)

```
## LINEAR REGRESSION

from sklearn.linear_model import LinearRegression

# Set universal seed for training
np.random.seed(0)

model1 = LinearRegression()

# Train model using the X_train and t_train.
model1.fit(X_train, t_train)

y1 = model1.predict(X_test)

test_score1 = model1.score(X_test,t_test)
print(f"Test score/Accuracy: {test_score1}")

# Evaluation metrics
print('\nEVALUATION METRICS:')
print('R2_score: ', r2_score(t_test, y1))
print('MaxErr: ', max_error(t_test,y1))
print('MAE: ', mean_absolute_error(t_test, y1))
print('MAPE: ', mean_absolute_percentage_error(t_test, y1))
print('MSE: ', mean_squared_error(t_test, y1))

#Plotting targets against predicted
plt.scatter(y1,t_test)
plt.xlabel("target")
plt.ylabel("predicted")
plt.title("Target against prediction for Linear Regression")
```

Result & Observation(continued)

```
Test score/Accuracy: 0.8437564137343283

EVALUATION METRICS:
R2_score:  0.8437564137343283
MaxErr:  144408.4179855144
MAE:  21782.460958898362
MAPE:  0.13676206496804927
MSE:  949653271.4733695
```



Result & Observation(continued)

```
## RIDGE REGRESSION

from sklearn.linear_model import Ridge

model2 = Ridge(alpha=10)

model2.fit(X_train,t_train)

test_score2 = model2.score(X_test,t_test)

y2= model2.predict(X_test)

print(f"Test score/Accuracy: {test_score2}")

# Evaluation metrics
print('\nEVALUATION METRICS:')
print('R2_score: ', r2_score(t_test, y2))
print('MaxErr: ', max_error(t_test,y2))
print('MAE: ', mean_absolute_error(t_test, y2))
print('MAPE: ', mean_absolute_percentage_error(t_test, y2))
print('MSE: ', mean_squared_error(t_test, y2))


#Plotting targets against predicted
plt.scatter(y2,t_test)
plt.xlabel("target")
plt.ylabel("predicted")
plt.title("Target against prediction for Ridge Regression")
```

Result & Observation(continued)

Test score/Accuracy: 0.8445160764819009

EVALUATION METRICS:

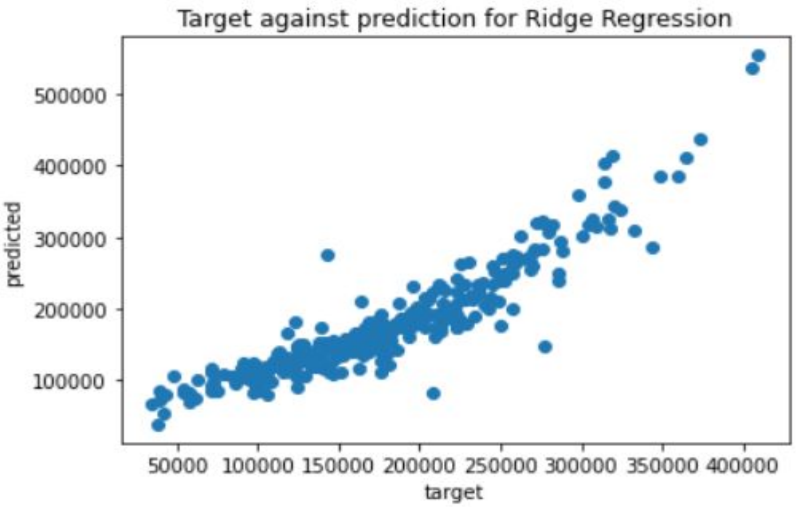
R2_score: 0.8445160764819009

MaxErr: 145322.99206565705

MAE: 21653.058038450068

MAPE: 0.13569852816307643

MSE: 945036018.1787478



Result & Observation(continued)

```
## LASSO REGRESSION

from sklearn.linear_model import Lasso

model3 = Lasso(alpha=0.1)

model3.fit(X_train,t_train)

test_score3 = model3.score(X_test,t_test)

y3= model3.predict(X_test)

print(f"Test score/Accuracy: {test_score3}")

# Evaluation metrics
print('\nEVALUATION METRICS:')
print('R2_score: ', r2_score(t_test, y3))
print('MaxErr: ', max_error(t_test,y3))
print('MAE: ', mean_absolute_error(t_test, y3))
print('MAPE: ', mean_absolute_percentage_error(t_test, y3))
print('MSE: ', mean_squared_error(t_test, y1))


#Plotting targets against predicted
plt.scatter(y3,t_test)
plt.xlabel("target")
plt.ylabel("predicted")
plt.title("Target against prediction for Lasso Regression")
```

Result & Observation(continued)

Test score/Accuracy: 0.8437576404122049

EVALUATION METRICS:

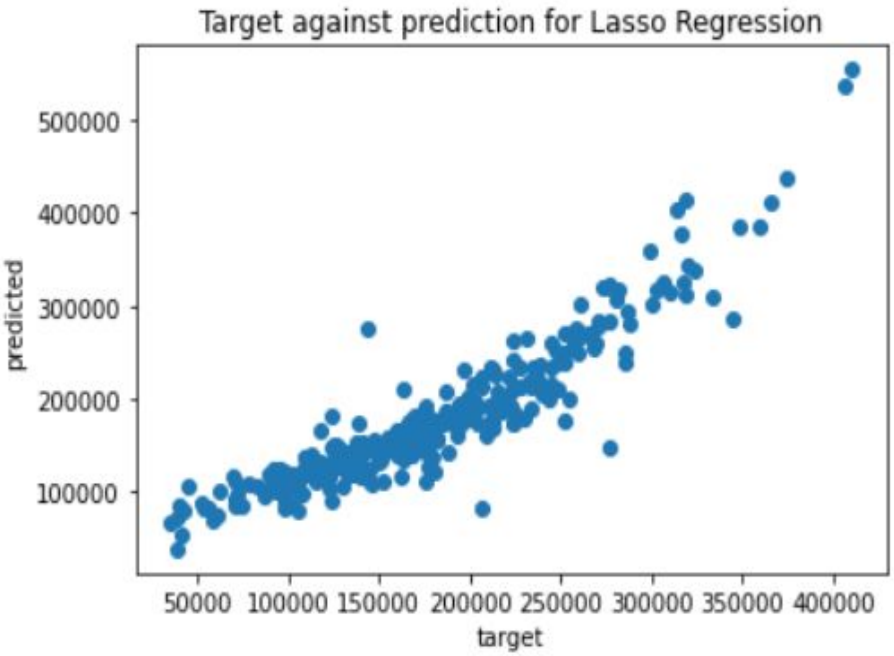
R2_score: 0.8437576404122049

MaxErr: 144409.60372044996

MAE: 21782.289561851198

MAPE: 0.1367608232780223

MSE: 949653271.4733695

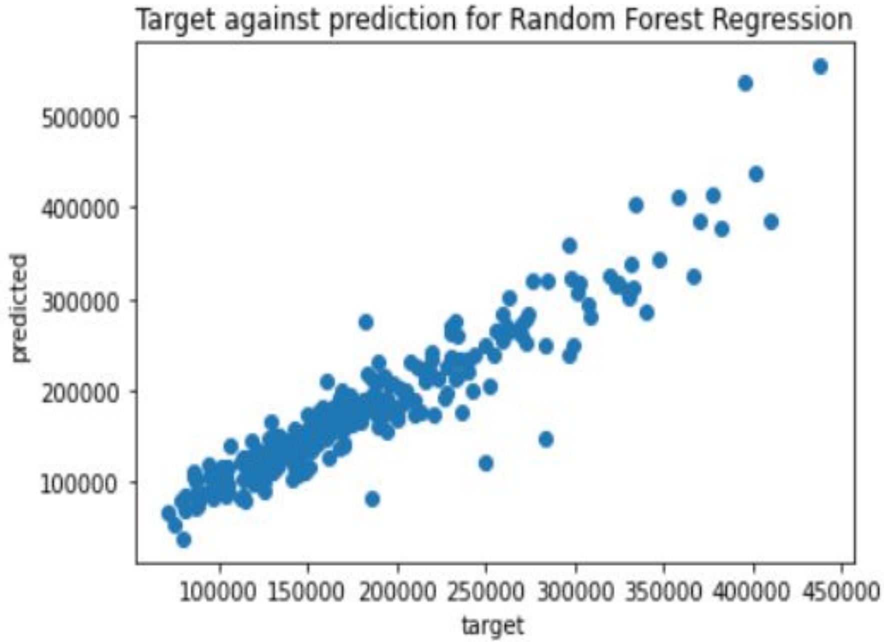


Result & Observation(continued)

```
## Random forest Regression

from sklearn.ensemble import RandomForestRegressor


model4 = RandomForestRegressor()

model4.fit(X_train,t_train)

test_score4 = model4.score(X_test,t_test)

y4= model4.predict(X_test)

print(f"Test score/Accuracy: {test_score4}")

# Evaluation metrics
print('\nEVALUATION METRICS:')
print('R2_score: ', r2_score(t_test, y4))
print('MaxErr: ', max_error(t_test,y4))
print('MAE: ', mean_absolute_error(t_test, y4))
print('MAPE: ', mean_absolute_percentage_error(t_test, y4))
print('MSE: ', mean_squared_error(t_test, y4))


#Plotting targets against predicted
plt.scatter(y4,t_test)
plt.xlabel("target")
plt.ylabel("predicted")
plt.title("Target against prediction for Random Forest Regression")
```

Result & Observation(continued)

```
Test score/Accuracy: 0.88616280426496

EVALUATION METRICS:
R2_score:  0.88616280426496
MaxErr:  142378.21000000002
MAE:  16940.69544520548
MAPE:  0.10547837816652159
MSE:  691905939.4944715
```



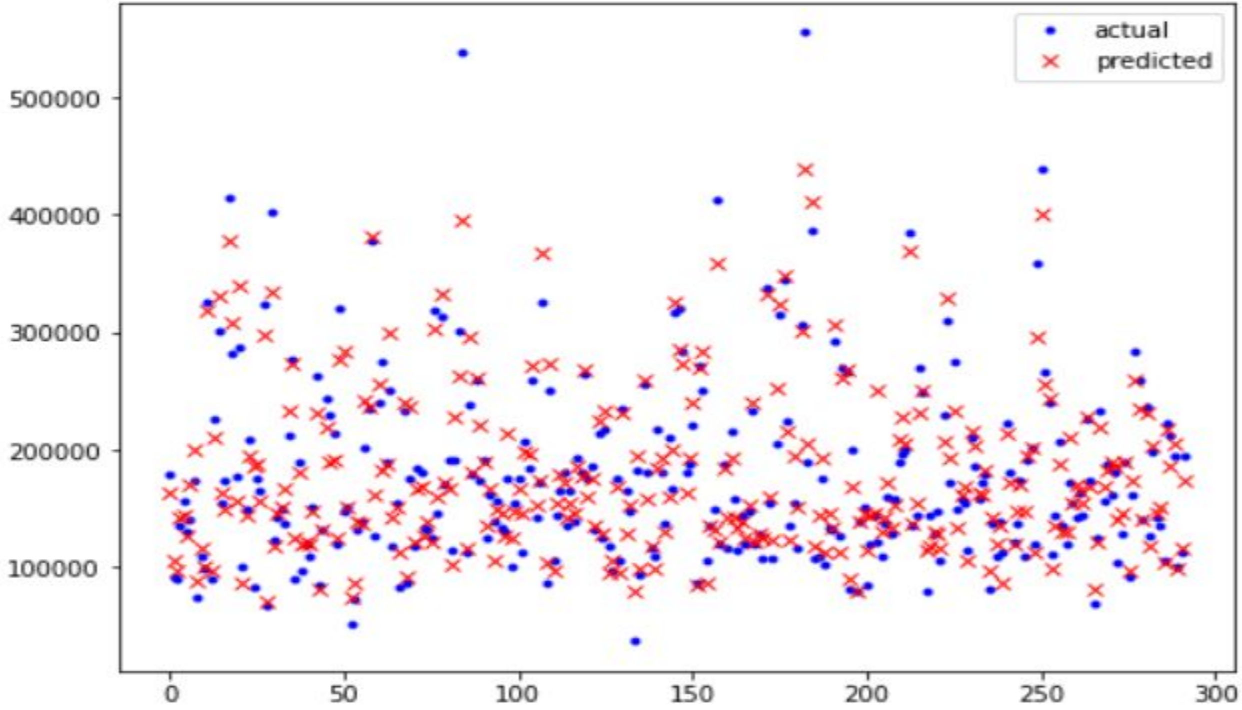
Result & Observation(continued)

From the previous comparisons, it can be seen that random forest performs the best with the highest accuracy score and the lowest evaluation scores among the models tested.

Now lets plot the predicted and the actual plot of the random forest model

```
# Finally looking at the predicted and the actual plot of the Random Forest Model
def targets_preds_plot(y4, t_test):
    plt.figure(figsize=(8,6))
    plt.plot(t_test.values, 'b.', label = 'actual')
    plt.plot(y4, 'rx', label = 'predicted')
    plt.legend()
    return

targets_preds_plot(y4, t_test)
```

Result & Observation(continued)



Conclusion / Future Work

The purpose of this project was fulfilled which was to check if random forest technique can predict house prices better than other regression models.

A lot more can be done with this dataset.

In future, I plan to explore explore the different classification techniques.

Feature selection and hyperparameter selection can also help develop a better prediction model in the future.

References

Abdul-Rahman, Shuzlina, et al. “Advanced Machine Learning Algorithms for House Price Prediction: Case Study in Kuala Lumpur.” *International Journal of Advanced Computer Science & Applications*, vol. 12, no. 12, 2021, https://doi.org/10.14569/IJACSA.2021.0121291.

Adetunji, Abigail Bola, et al. “House Price Prediction Using Random Forest Machine Learning Technique.” *Procedia Computer Science*, vol. 199, 2022, pp. 806–13, https://doi.org/10.1016/j.procs.2022.01.100.

Truong, Quang, et al. “Housing Price Prediction via Improved Machine Learning Techniques.” *Procedia Computer Science*, vol. 174, 2020, pp. 433–42, https://doi.org/10.1016/j.procs.2020.06.111.