Data Analysis, Machine Learning and

A.I. using python

Used bike price prediction using XGBRegressor

Submitted to:
Mr. Bandenawaaz Bagwan

Submitted by:
Utkarsh Balooni

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my instructor Mr. Bandenawaaz Bagwan for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I would also want to thank my parents and friends who helped me in finalizing this project within a limited time frame.

Introduction

Determining the listed price of a used bike is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used bike based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different bikes across cities. The results show that Random Forest model and XGBRegressor yield the most accurate results. We shall be using XGBRegressor for training the data for the flask app.

Data used

The dataset used for this project has been obtained from www.kaggle.com. It contains information about approx 32000 used bikes scraped from www.droom.in. It includes features like power, kilometers driven, Age of the bike etc.

	bike_name	price	city	kms_driven	owner	age	power	brand
0	TVS Star City Plus Dual Tone 110cc	35000.0	Ahmedabad	17654.0	First Owner	3.0	110.0	TVS
1	Royal Enfield Classic 350cc	119900.0	Delhi	11000.0	First Owner	4.0	350.0	Royal Enfield
2	Triumph Daytona 675R	600000.0	Delhi	110.0	First Owner	8.0	675.0	Triumph
3	TVS Apache RTR 180cc	65000.0	Bangalore	16329.0	First Owner	4.0	180.0	TVS
4	Yamaha FZ S V 2.0 150cc-Ltd. Edition	80000.0	Bangalore	10000.0	First Owner	3.0	150.0	Yamaha

Algorithm used (XGBRegressor)

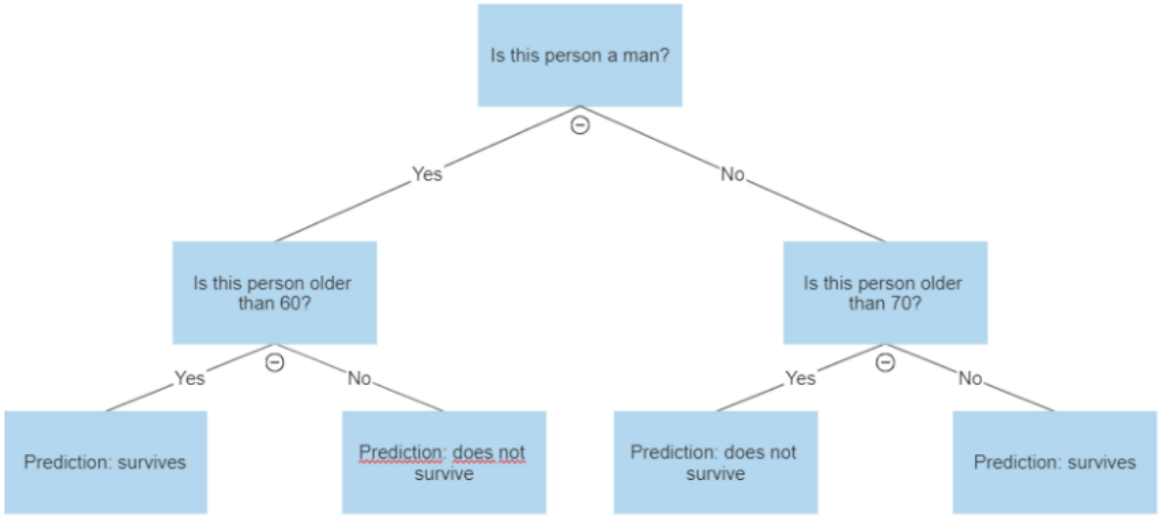
1. Introduction

XGBoost stands for eXtreme Gradient Boosting and it's an open-source implementation of the gradient boosted trees algorithm. It provides high prediction power and ease of use. It is a supervised learning algorithm that can be used

for regression or classification tasks. To understand XGBoost we must first understand Decision Trees and Gradient Boosting.

2. Decision Trees

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter.A decision tree has this name because of its visual shape, which looks like a tree, with a root and many nodes and leaves. A simple decision tree for the prediction of survival on the titanic would look like this:



3. Gradient Boosting

Boosting is an ensemble method, meaning it's a way of combining predictions from several models into one. It does that by taking each predictor sequentially and modelling it based on its predecessor's error (giving more weight to predictors that perform better):

1. Fit a first model using the original data.
2. Fit a second model using the residuals of the first model.

3. Create a third model using the sum of models 1 and 2.

Gradient boosting is a specific type of boosting, called like that because it minimises the loss function using a gradient descent algorithm.

3. XGBoost working

XGBoost is a gradient boosting algorithm that uses decision trees as its "weak" predictors. Its implementation was specifically engineered for optimal performance and speed. XGBoost performs quite well for structured, tabular data. If you are dealing with non-structured data such as images, neural networks are usually a better option.

4. Hyperparameters for XGBoost

- **booster:** booster is the boosting algorithm, for which there are 3 options: gbtree, gblinear or dart . The default option is gbtree.
- **reg_alpha and reg_lambda:** reg_alpha and reg_lambda are L1 and L2 regularisation terms, respectively. The greater these numbers, the more conservative the model becomes.
- **max_depth:** max_depth sets the maximum depth of the decision trees. The greater this number, the less conservative the model becomes.
- **subsample:** subsample is the size of the sample ratio to be used when training the predictors.
- **num_estimators:** num_estimators sets the number of boosting rounds, which equals setting the number of boosted trees to use.

Procedure

1. **Step-1:** Read the dataset.

```
used_bikes=pd.read_csv('Used_Bikes.csv')
```

2. **Step-2:** Select the relevant features to be used for training the model.

```
X_cols=['kms_driven','owner','age','power','brand']
y_cols=['new_price']
```

3. **Step-3:** Encode the categorical variables.

```
X=used_bikes[X_cols]
y=used_bikes[y_cols]

encoder=OrdinalEncoder()
X_encoded=X.copy()

X_encoded[obj_cols]=encoder.fit_transform(X[obj_cols])
```

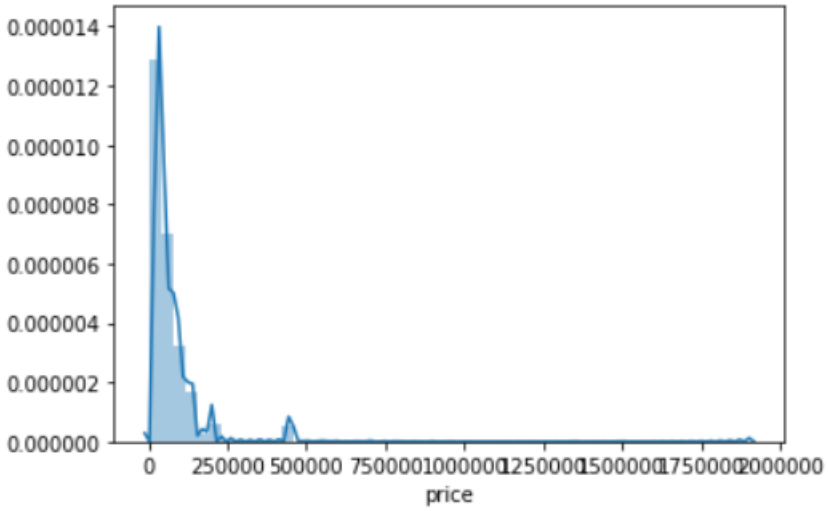
4. **Step-4:** Fit the model.

```
xgb=XGBRegressor()
xgb.fit(X_encoded.values,y.values)
```

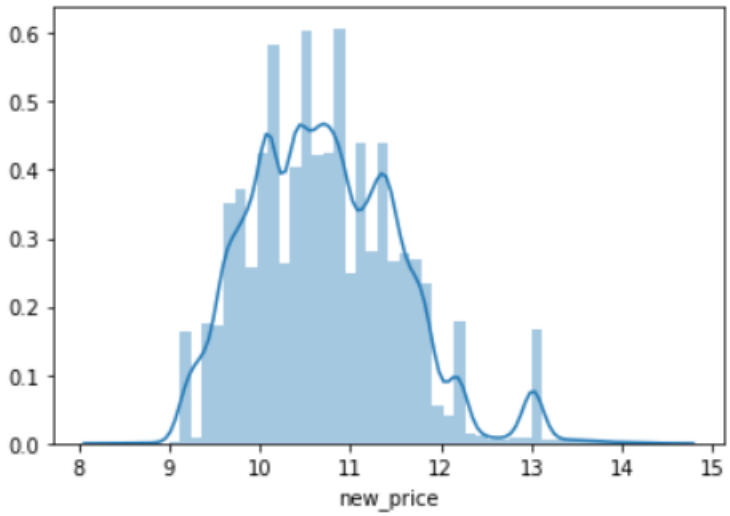
5. **Step-5:** Check the accuracy score for the model.

```
cross_validate(xgb,X_new,y,scoring='r2')['test_score'].mean()
```

Note: While pre-processing the data we found out that the output variable,i.e price has a very skewed distribution.



Hence to reduce the skewness we transformed it and used the log of the values of price instead as the output variable. The distribution for the new price variable can be considered approximately normal.



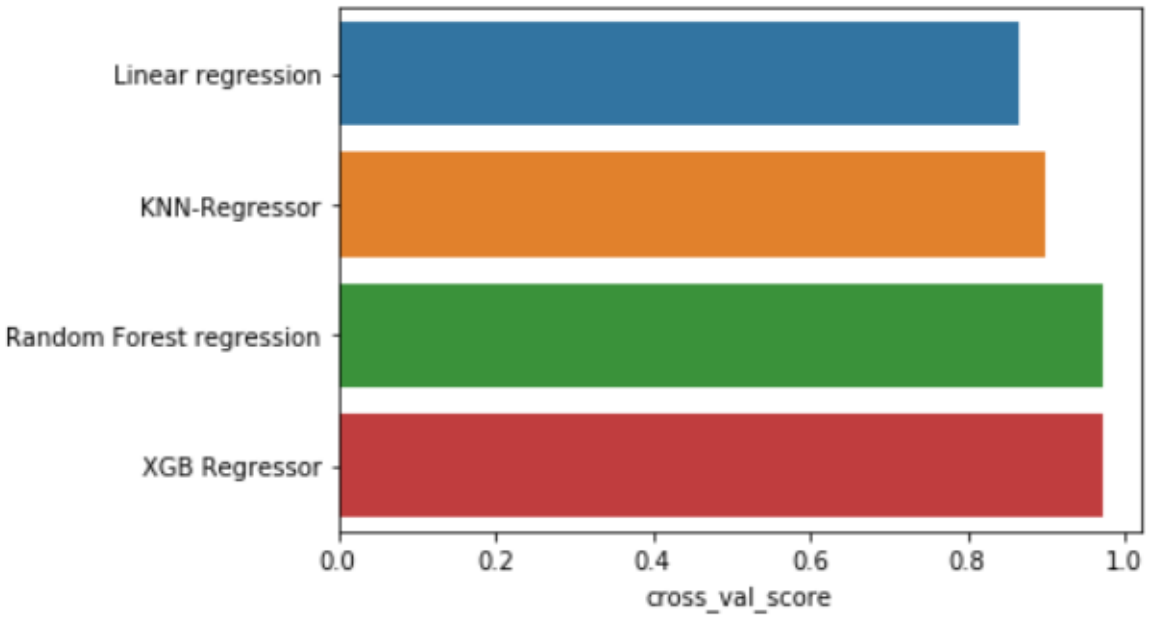
Results

We trained 4 different models using the data. These were:

- Linear Regressor
- KNN-Regressor
- Random Forest Regressor
- XGBRegressor

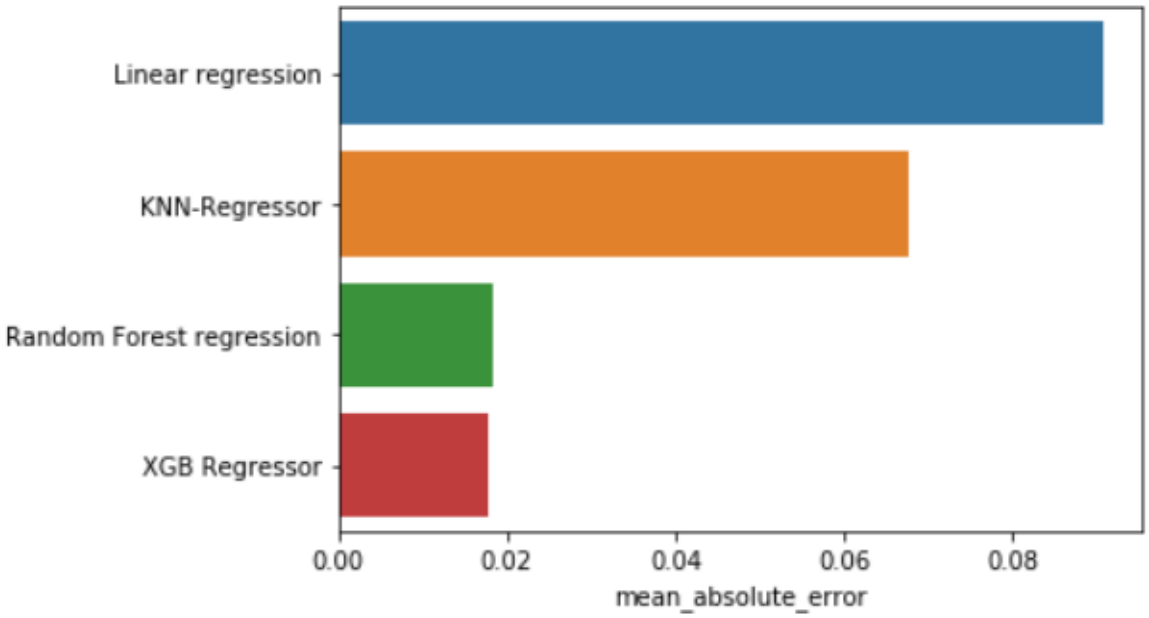
Mean cross-validation score calculated for each of the models are as follows:

```
{'Linear regression': 0.8648210888093522,
 'KNN-Regressor': 0.899556283432366,
 'Random Forest regression': 0.9730493309521469,
 'XGB Regressor': 0.9736650490214291}
```



Mean absolute error calculated for each of the models are as follows:

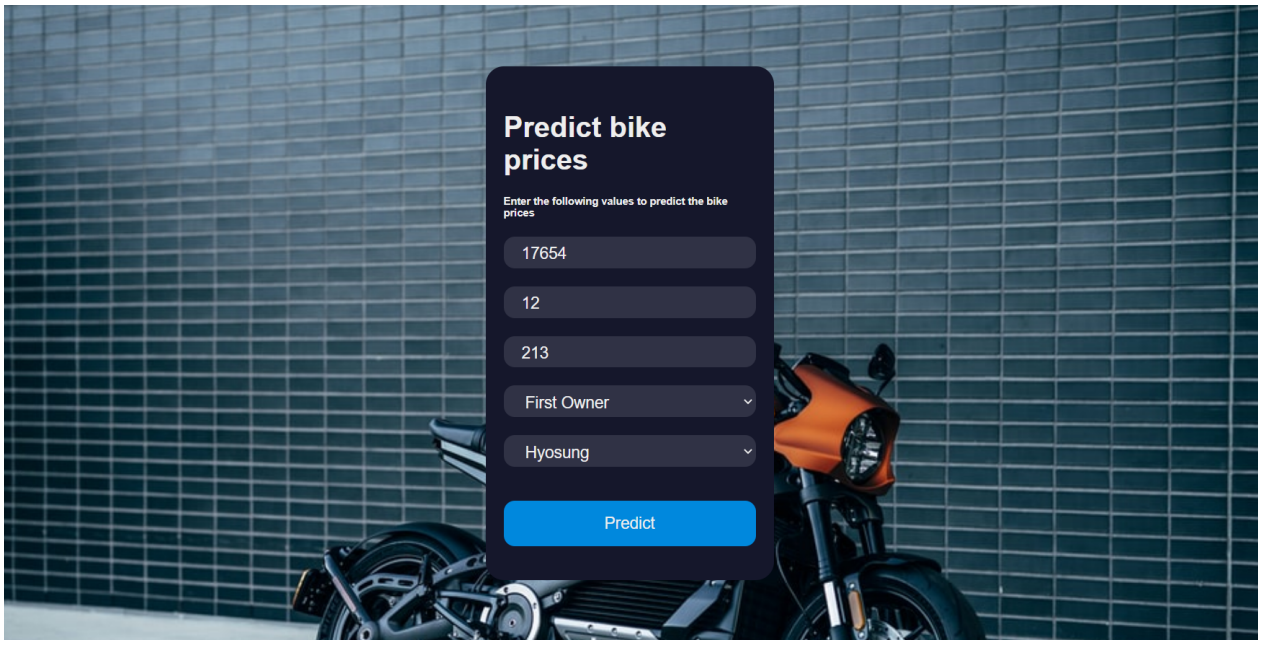
```
{'Linear regression': 0.09082994450450323,
 'KNN-Regressor': 0.0676384109539986,
 'Random Forest regression': 0.01811714151866481,
 'XGB Regressor': 0.017742222552566033}
```



Conclusion

Hence, we can conclude that XGBRegressor gives the best accuracy score (0.9736) while linear regressor gives the worst (0.8648).

Flask app



Price of bike is : 51190.66015625

Drawbacks

- The model created does not take into account economic factors like inflation which can result in error in the results.

References

- https://towardsdatascience.com/xgboost-theory-and-practice
- https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm..
- https://www.kaggle.com/saisaathvik/used-bikes-prices-in-india