BANGALORE HOUSE PRICE PREDICTION

A data science project from scratch

BY

Heba Mohamed

PROBLEM STAMENT

What are the things that a potential home buyer considers before purchasing a house?

The location, the size of the property, vicinity to offices, schools, parks, restaurants, hospitals or the stereotypical white picket fence?

What about the most important factor — the price?

PROPOSED SOLUTION

I provided a study methodology that employs a range of algorithms (Logistic Regression, K-Folds, and DNN) to assess the dataset's most vital parts or attributes and compare their accuracy.

METHODOLOGY

- LOAD DATA
- DATA CLEANING
- DATA VISUALIZATION
- DATA PREPARING
- DATA MODELING
  - Building
  - Training
  - Evaluating
  - Testing

DATASET BEFORE CLEANING

Shape = (13320, 9)

```
df = pd.read_csv('Bengaluru_House_Data.csv')
df.head()
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

CLEANING PROCESSES

Handle null values

Feature Engineering

- Add new feature (bedrooms)
- Fix (total_sqft) feature
- Add new feature called (price_per_sqft)

Dimensionality reduction for categorical feature (location)

Outlier removal

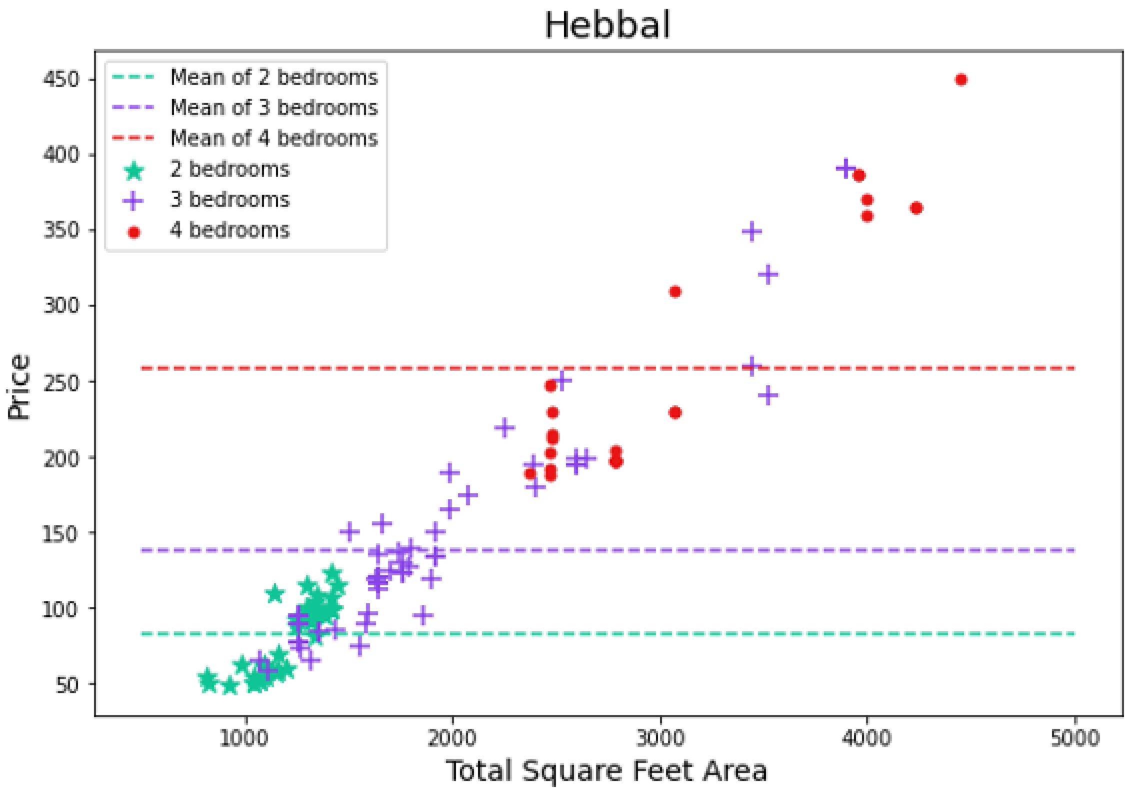
- Outlier removal from (price_per_sqft) feature according to business logic
- Outlier removal from (price_per_sqft) feature according to std and mean
- Outlier removal from (bedroom) feature according to business logic
- Outlier removal from (bath) feature

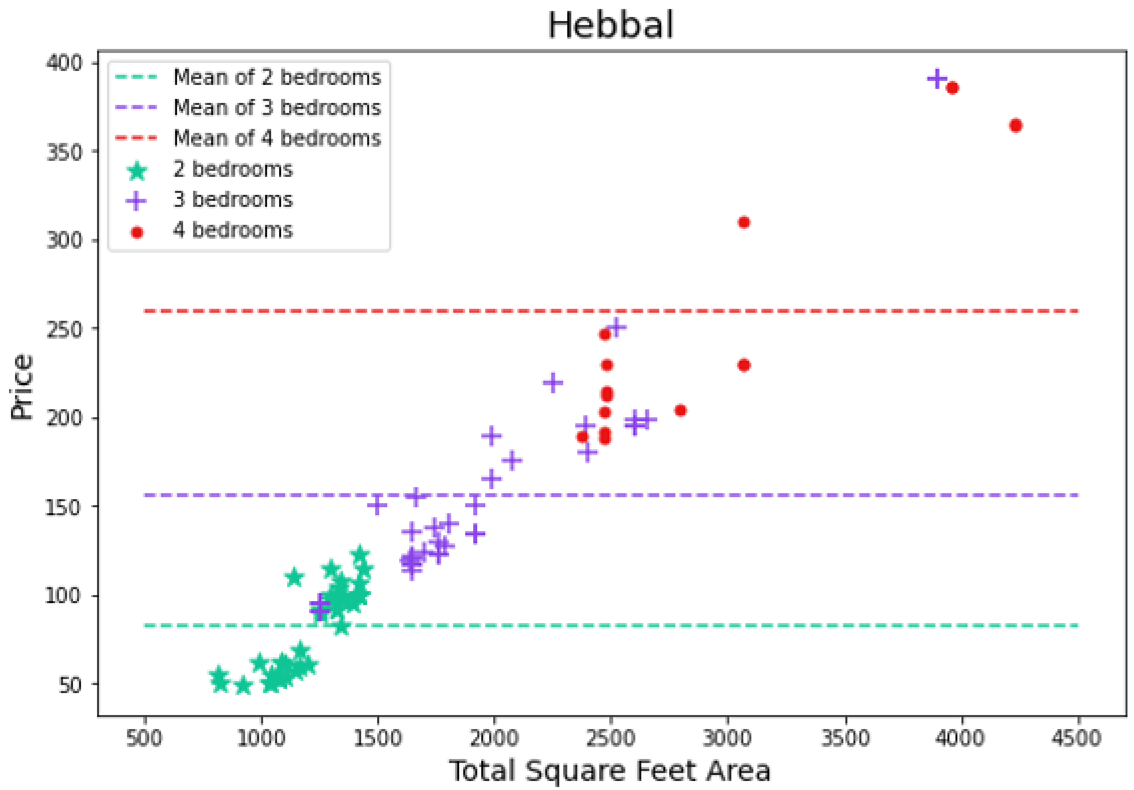
DATASET AFTER CLEANING

Shape = (7268, 244)

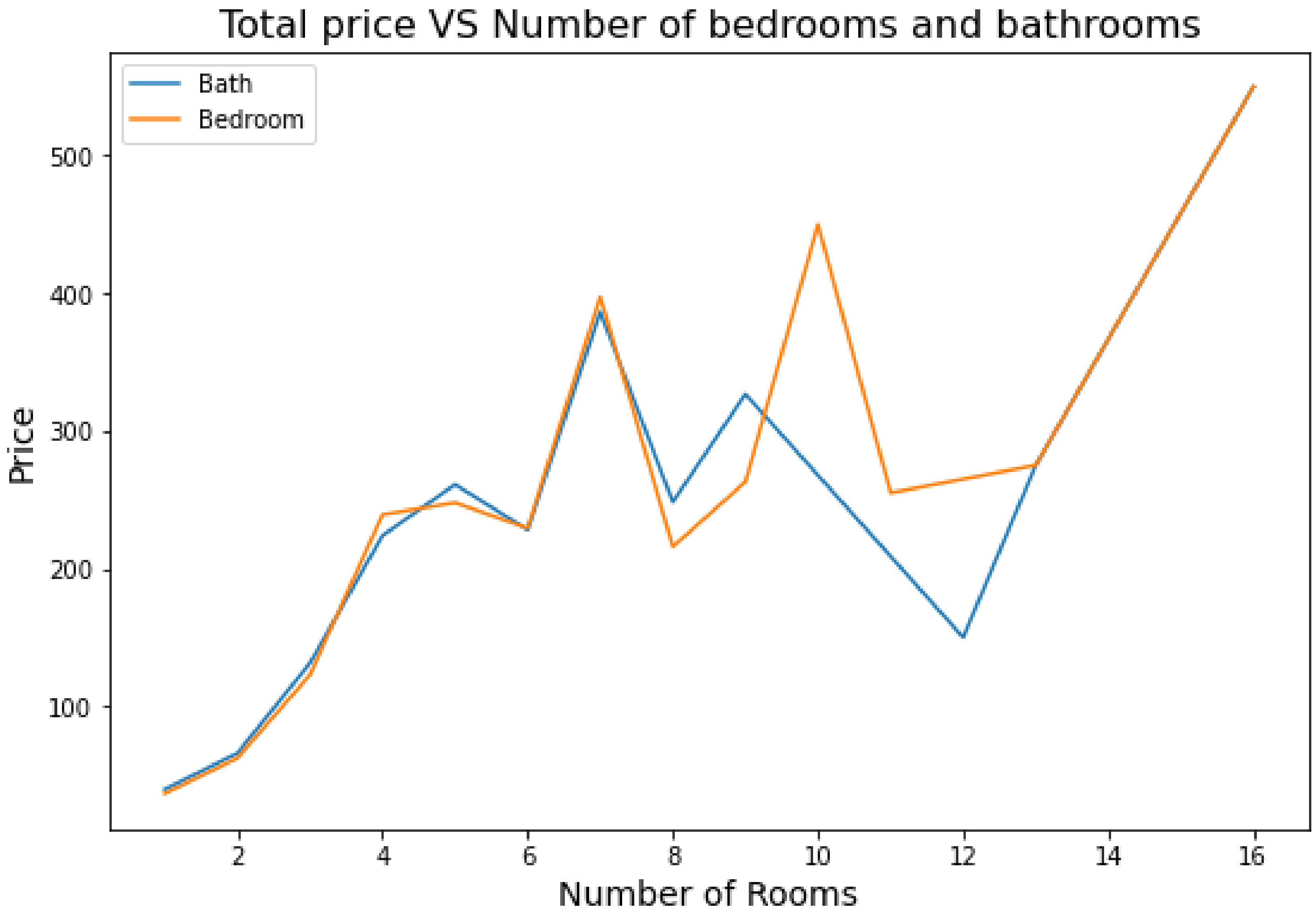
	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

Scatter chart for (bedroom) feature before and after outliers' removal

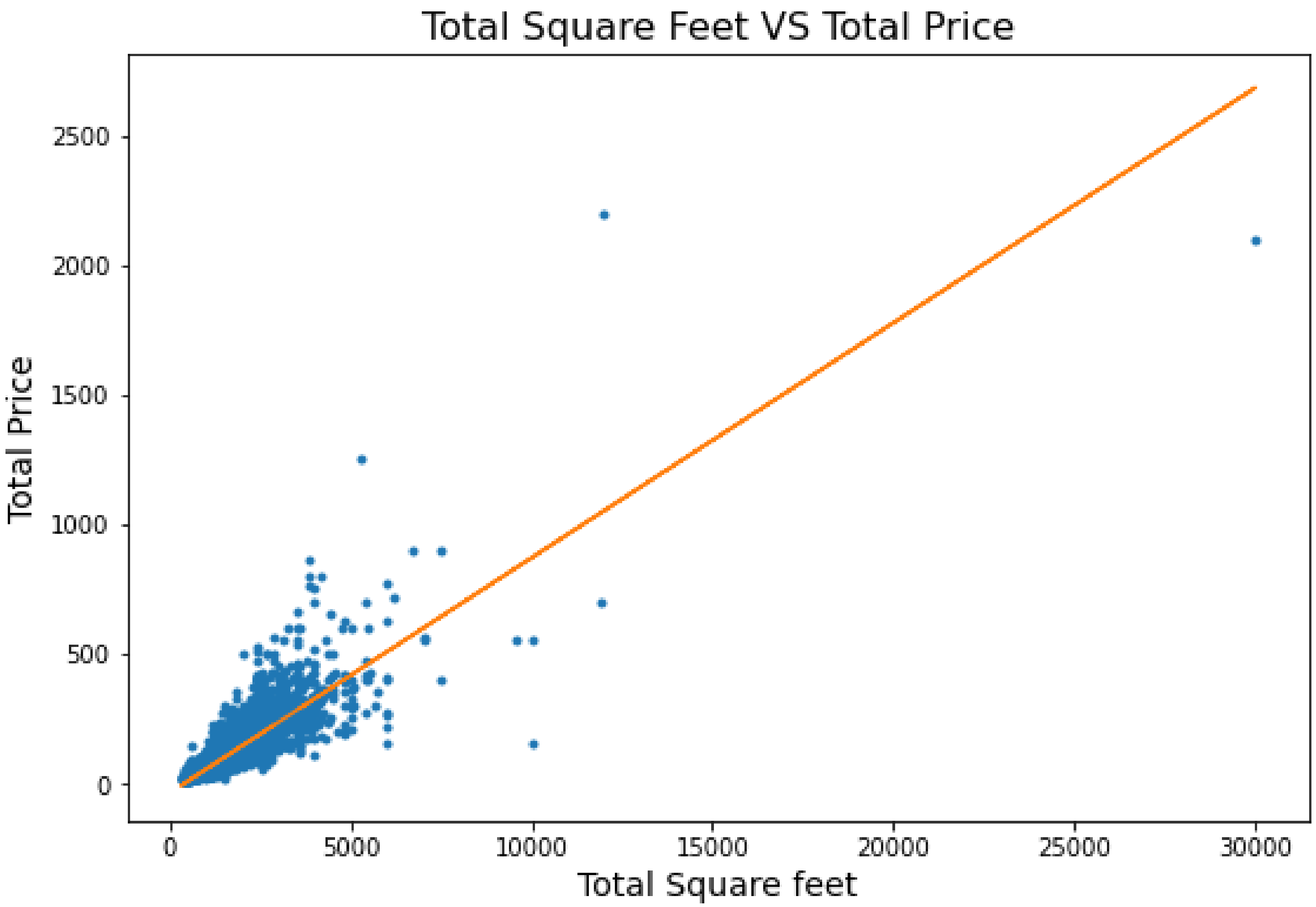




Relation between Total price vs Number of bedrooms and bathrooms.



Trend chart for Total square feet vs Total price.



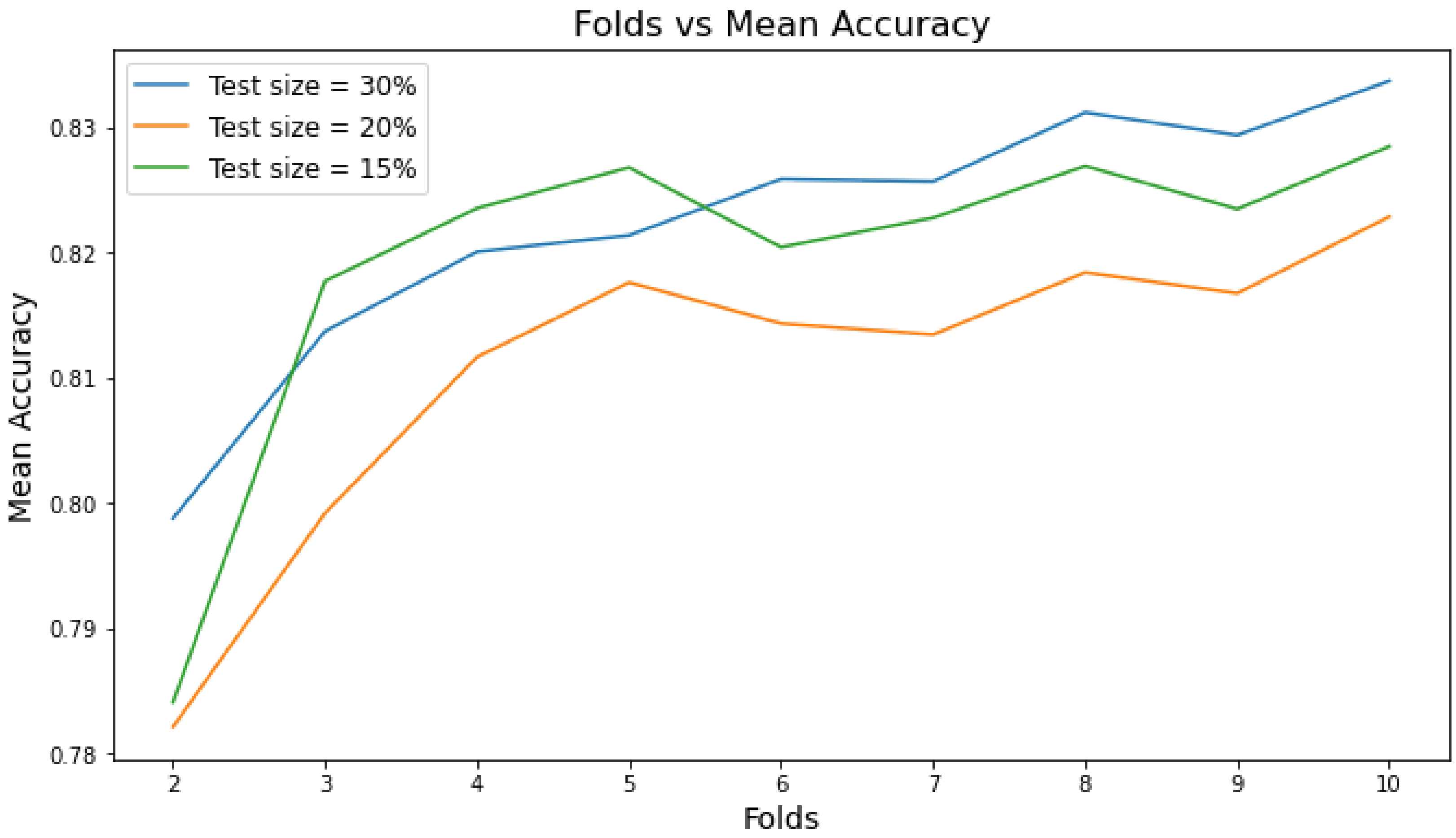
Multivariate Linear Regression model

This model give accuracy = 77.2% which is pretty low, so I will go and try the regression model using K-Folds.

Searching for Best Regression Model Using K-Fold Cross Validation

- Shuffle the dataset randomly with test size= [0.3,0.2,0.15].
- Split the dataset into k-folds groups folds=[2,3,4,5,6,7,8,9,10].

- Best Accuracy = 83.0%
- Test size = 30.0%
- Folds = 8
- The accuracy of the model is pretty good, but I will improve it using deep learning technique.



Deep Learning Model

Hyperparameter	values
Test size	20%
Number of hidden layers	10
Hidden layers size	[100,90,80,70,60,50,40,30,20,10]
Activation function	ReLU
Optimizer	Adam
learning rate	0.0001
Number of epochs	500
Callbacks	Model Check Point
Metric	Mean Square Error

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 110)               26840
_________________________________________________________________
dense_1 (Dense)              (None, 100)               11100
_________________________________________________________________
dense_2 (Dense)              (None, 90)                9090
_________________________________________________________________
dense_3 (Dense)              (None, 80)                7280
_________________________________________________________________
dense_4 (Dense)              (None, 70)                5670
_________________________________________________________________
dense_5 (Dense)              (None, 60)                4260
_________________________________________________________________
dense_6 (Dense)              (None, 50)                3050
_________________________________________________________________
dense_7 (Dense)              (None, 40)                2040
_________________________________________________________________
dense_8 (Dense)              (None, 30)                1230
_________________________________________________________________
dense_9 (Dense)              (None, 20)                620
_________________________________________________________________
dense_10 (Dense)             (None, 10)                210
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 11
=================================================================
Total params: 71,401
Trainable params: 71,401
Non-trainable params: 0
_________________________________________________________________
```

In [77]:

```
n_epochs = 500

hist = ... model.fit(
```

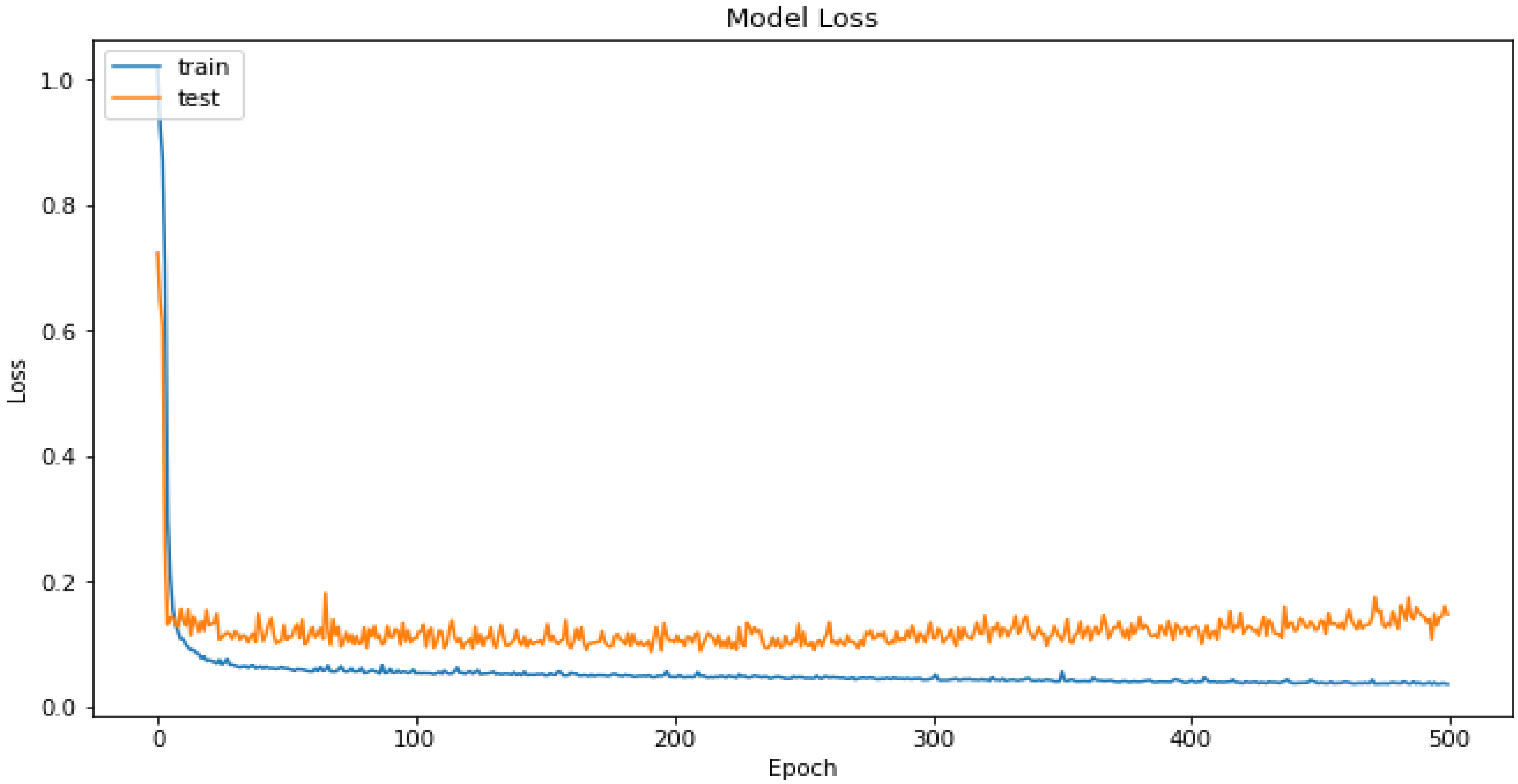
Training Deep Learning Model

with 500 epochs

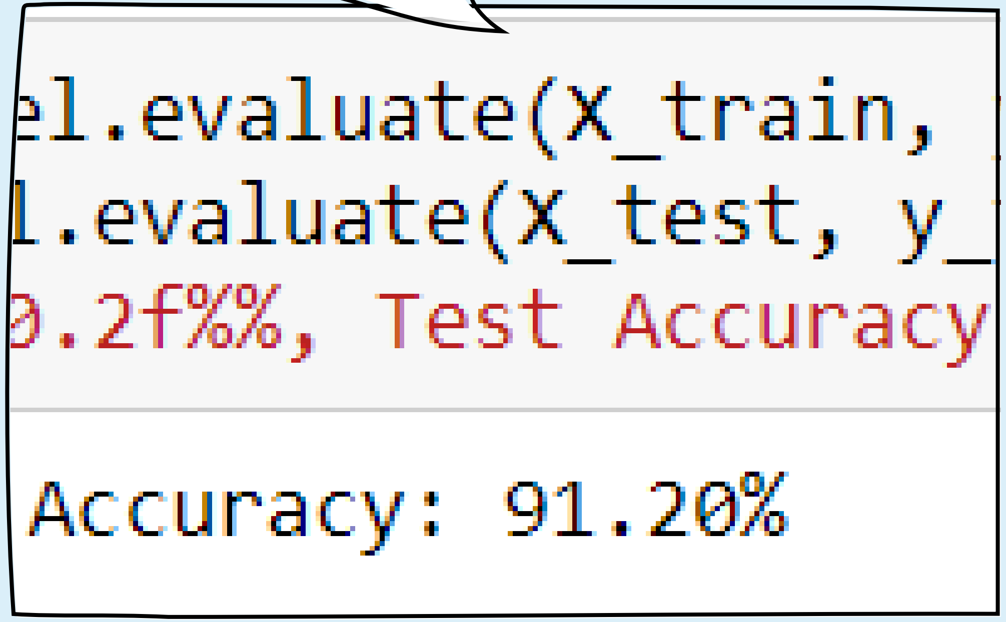
```
194/194 [==============================] - 0s 2ms/step - loss: 1.0155 - mse: 1.0155 - val_loss: 0.7235 - val_mse: 0.7235
Epoch 2/500
194/194 [==============================] - 0s 2ms/step - loss: 0.9250 - mse: 0.9250 - val_loss: 0.6476 - val_mse: 0.6476
Epoch 3/500
194/194 [==============================] - 0s 2ms/step - loss: 0.8725 - mse: 0.8725 - val_loss: 0.6063 - val_mse: 0.6063
Epoch 4/500
194/194 [==============================] - 0s 2ms/step - loss: 0.7035 - mse: 0.7035 - val_loss: 0.2498 - val_mse: 0.2498
Epoch 5/500
194/194 [==============================] - 0s 1ms/step - loss: 0.3011 - mse: 0.3011 - val_loss: 0.1316 - val_mse: 0.1316
Epoch 6/500
194/194 [==============================] - 0s 1ms/step - loss: 0.2079 - mse: 0.2079 - val_loss: 0.1432 - val_mse: 0.1432
Epoch 7/500
194/194 [==============================] - 0s 1ms/step - loss: 0.1563 - mse: 0.1563 - val_loss: 0.1427 - val_mse: 0.1427
Epoch 8/500
194/194 [==============================] - 0s 1ms/step - loss: 0.1319 - mse: 0.1319 - val_loss: 0.1323 - val_mse: 0.1323
Epoch 9/500
194/194 [==============================] - 0s 2ms/step - loss: 0.1195 - mse: 0.1195 - val_loss: 0.1277 - val_mse: 0.1277
Epoch 10/500
194/194 [==============================] - 0s 1ms/step - loss: 0.1084 - mse: 0.1084 - val_loss: 0.1567 - val_mse: 0.1567
Epoch 11/500
194/194 [==============================] - 0s 1ms/step - loss: 0.1082 - mse: 0.1082 - val_loss: 0.1372 - val_mse: 0.1372
Epoch 12/500
194/194 [==============================] - 0s 1ms/step - loss: 0.0997 - mse: 0.0997 - val_loss: 0.1297 - val_mse: 0.1297
```

Deep learning history

(training and validation losses vs number of epochs) function.



The best model was preserved during training as a consequence of the use of the model check point API, I utilized it for testing and documenting the final findings.



Deep learning Accuracy

Model	Accuracy
Multivariate Regression	77.17%
K-Fold Regression	83.0%
Deep Learning	91.20%

Conclusion

Therefore, it is clear that the third model that was built using deep learning is the most efficient among the three, with an accuracy of 91.20%.