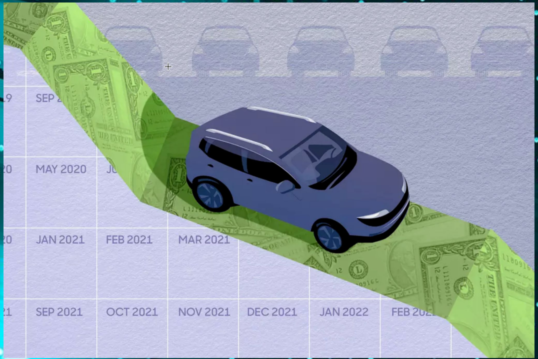
Car Price Prediction

By: Adkeme Berhe

Today's Agenda

1. Background
2. Business problem
3. Data Sources
4. Data Preparation
5. Modeling
6. Evaluation



Business Problem

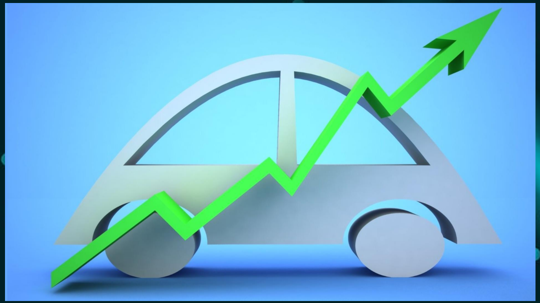
- Due to an ongoing chip shortage and increased cost in raw materials the U.S average price for a new car was up 6%

- Ahead of 2023's first quarter, Group 11 I have been tasked with creating a Price Prediction model to help determine car prices.

Data Sources

- Obtained from Kaggle at https://www.kaggle.com/datasets/deepcontractor/car-price-prediction-challenge/discussion/367511

- **19,237** rows, **18** columns. **(2.2mb)**
  Most notable Car features
  - Price
  - Manufacturer
  - Model
  - Fuel Type
  - Production Year



Data Preparation

- Before any cleaning or pre processing model ran at a 6% accuracy
- Went through and checked for NULL Values
- After further analyzing the data, 313 duplicate values were fund and dropped.
- With prices ranging from $0 to 2.38 billion we made it more practical, removing <1,000 and >100,000
- Mileage included KM unit which needed to be removed for numerical reasons
- Missing Levy feature resulted in dropping

FROM: 19,237 —> TO: 8,027 rows

(75%-25% split) of our remaining 8,027 rows

FROM: 18 —>17 columns

In [132]:
```
cars_updated = dup.copy()
updated1 = cars_updated['Price'] < 10000
updated1.head()
updated2 = cars_updated['Price'] > 100000
updated2.head()
cars_updated.loc[updated1, 'Price'] = np.nan
cars_updated.loc[updated2, 'Price'] = np.nan
cars_updated.head()
```

Out[132]:

	ID	Price	Levy	Manufacturer	Model	ProdYear	Category	LeatherInterior	FuelType	EngineVolume	Mileage	Cylinders	GearBoxType	DriveWheel
0	45654403	13328.0	1399	LEXUS	RX 450	2010	Jeep	Yes	Hybrid	3.5	186005 km	6.0	Automatic	4x
1	44731507	16621.0	1018	CHEVROLET	Equinox	2011	Jeep	No	Petrol	3	192000 km	6.0	Tiptronic	4x
2	45774419	NaN	-	HONDA	FIT	2006	Hatchback	No	Petrol	1.3	200000 km	4.0	Variator	Fro
3	45769185	NaN	862	FORD	Escape	2011	Jeep	Yes	Hybrid	2.5	168966 km	4.0	Automatic	4x
4	45809263	11726.0	446	HONDA	FIT	2014	Hatchback	Yes	Petrol	1.3	91901 km	4.0	Automatic	Fro

In [133]:
```
cars_updated.isna().sum().sum()
```

Out[133]: 7546

In [134]:
```
cars_updated.dropna(inplace=True)
```

202201-chip-s....webp 072919-RL-Av....webp 072919-RL-Av....webp carprediction.jpeg download (3).jpeg

```
# To remove the "km" in the Mileage
cars_updated['Mileage'] = cars_updated['Mileage'].str.replace("km",'', regex=True)
cars_updated['Mileage'] = pd.to_numeric(cars_updated['Mileage'], errors='coerce')
cars_updated
```

	ID	Price	Levy	Manufacturer	Model	ProdYear	Category	LeatherInterior	FuelType	EngineVolume	Mileage	Cylinders	GearBoxType	DriveW
0	45654403	13328.0	1399	LEXUS	RX 450	2010	Jeep	Yes	Hybrid	3.5	186005	6.0	Automatic	
1	44731507	16621.0	1018	CHEVROLET	Equinox	2011	Jeep	No	Petrol	3	192000	6.0	Tiptronic	
4	45809263	11726.0	446	HONDA	FIT	2014	Hatchback	Yes	Petrol	1.3	91901	4.0	Automatic	
5	45802912	39493.0	891	HYUNDAI	Santa FE	2016	Jeep	Yes	Diesel	2	160931	4.0	Automatic	
9	45756839	26657.0	-	LEXUS	RX 350	2007	Jeep	Yes	Petrol	3.5	128500	6.0	Automatic	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
19222	45732720	10036.0	639	HYUNDAI	Sonata	2014	Sedan	Yes	LPG	2	735970	4.0	Automatic	
19224	45768089	19130.0	-	KIA	Optima ex	2014	Sedan	Yes	Petrol	2.4	35800	4.0	Automatic	
19227	45769427	29793.0	1053	MERCEDES-BENZ	E 350	2014	Sedan	Yes	Diesel	3.5	219030	6.0	Automatic	
19233	45778856	15681.0	831	HYUNDAI	Sonata	2011	Sedan	Yes	Petrol	2.4	161600	4.0	Tiptronic	
19234	45804997	26108.0	836	HYUNDAI	Tucson	2010	Jeep	Yes	Diesel	2	116365	4.0	Automatic	

11378 rows × 18 columns

In [262]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=5)
```

Model

```python
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.neighbors import KNeighborsRegressor
```

```python
In [261]: x = cars_f.drop(['Price', 'Color'], axis=1).values
          y = cars_f['Price'].values
          x = MinMaxScaler().fit_transform(x)

In [262]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=5)

In [263]: #lr  = LinearRegression()
          knn = KNeighborsRegressor(n_neighbors=8)
          dt  = DecisionTreeRegressor(max_depth = 5)
          rf  = RandomForestRegressor(n_estimators=100, max_features= 7)

          regressors = [ ('K Nearest Neighbours', knn),('Decision Tree', dt), ('Random Forest', rf)]

In [264]: for regressor_name, regressor in regressors:

              regressor.fit(x_train, y_train)

              y_pred = regressor.predict(x_test)
              accuracy = round(r2_score(y_test,y_pred),3)*100

              print('{:s} : {:.0f} %'.format(regressor_name, accuracy))
              plt.rcParams["figure.figsize"] = (20,8)
              plt.bar(regressor_name,accuracy)
```

knn=8 neighbors

dt= depth of 5

rf= 7 features, 100 estimators

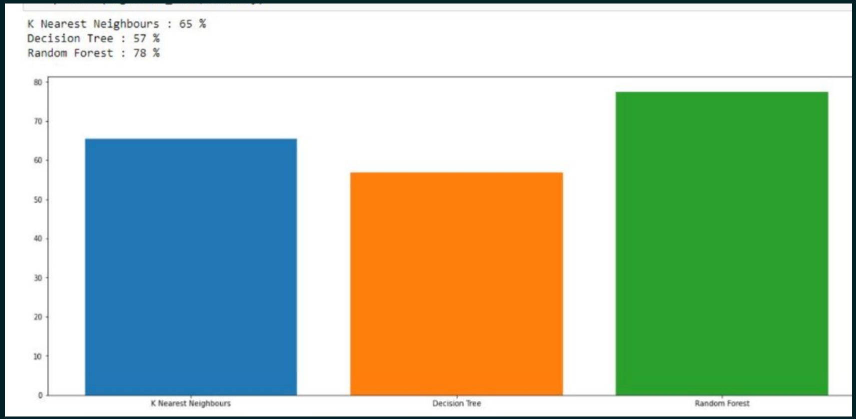
Stored in regressors, fitted our testing

Printed our prediction & accuracy

Created a bar graph for depiction

Evaluation

Model Type	KNN	Decision Tree	Random Forest
Parameters	N neighbors value	Max Depth	Estimator & max features
Values	8	5	100 and 7
Accuracy %	65%	48%	77%



Recommendations & Lessons Learned

While we are satisfied with our Random Forest Model, more time may have allowed us the chance to further increase our model accuracy

Buyer and seller discretion is advised due to accuracy level being moderate.

Data preparation is the most tedious (and annoying) part of everything. Its importance cant be understated.

Dont forget to consider business value and ROI for companies

Future work

- Currently no plans to extend project any further.
- I will utilize my project as a resume portfolio piece .

Adkeme Berhe
https://www.linkedin.com/in/adkeme-berhe-2bb456240/

Thank you for your time !