Used Vehicles

Alex Timperman, Zehra Amir , Michelle Fulgencio, Shazir Burney

Objectives:

- Data preparation by cleansing and removing unwanted variables.
- Data visualization using different libraries such as matplotlib and seaborn.
- Data processing using Linear Regression and Machine Learning algorithms.
- Identifying what features contribute to selling a used car quickly.

Business Questions

- It is necessary for car sellers to sell used cars quickly because
  - they need to use the money to buy new model cars
  - keeping used cars for too long lowers its value.
  - It can be a waste of resources as it occupies space and has high maintenance costs.

- While analyzing the color, body type, model, price of the car, what are the chances a car can be sold as quickly as possible?

About the Data:

This dataset is about used cars. After cleaning the dataset we decided to work with a few variables:

(variables listed with their type)

- Transmission: Object
- Color: Object
- Odometer value: Int
- Year produced: Int
- Engine fuel: Object
- Engine has gas: Bool
- Engine type: Object
- Body type: Object
- Engine capacity: Float
- Has warranty: Bool
- State: Object
- Drive train: Object
- Price in USD: Float
- Is exchangeable: Bool
- Number of photos: Int
- Up counter: Int
- Duration listed: Int

- We are predicting the duration listed which would tell us how long it took for the cars to be sold.
- There were 38530 records of the variables.
- The rows that had null values were removed from the dataset (only 10).

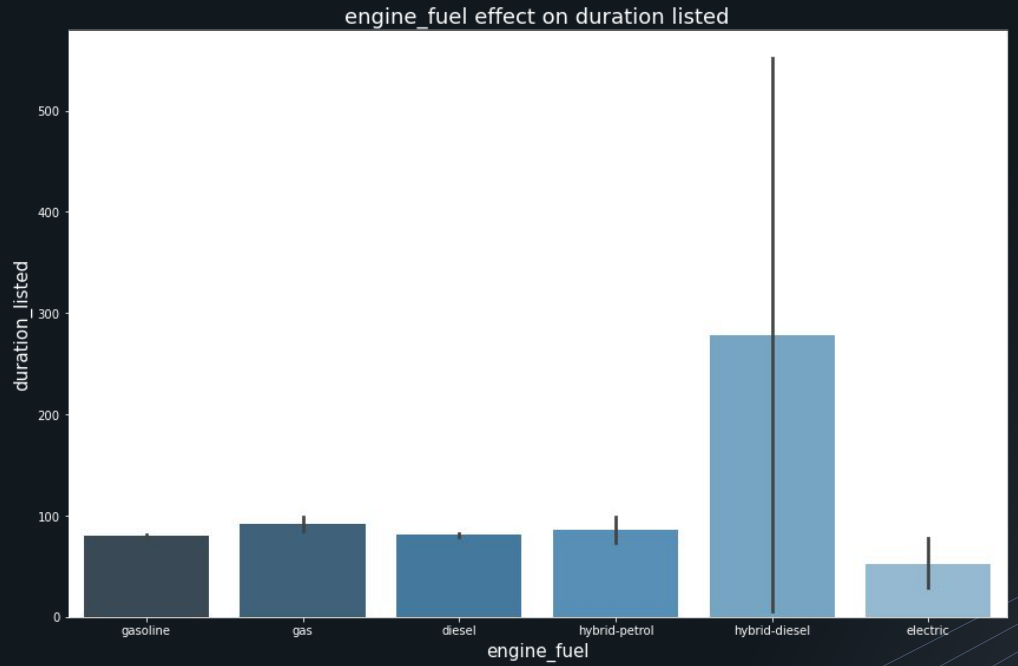
Pandas was used to read the data and finally have it look like this:

	transmission	color	odometer_value	year_produced	engine_fuel	engine_has_gas	engine_type	engine_capacity	body_type	has_warranty	state
0	automatic	silver	190000	2010	gasoline	False	gasoline	2.5	universal	False	owned
1	automatic	blue	290000	2002	gasoline	False	gasoline	3.0	universal	False	owned
2	automatic	red	402000	2001	gasoline	False	gasoline	2.5	suv	False	owned
3	mechanical	blue	10000	1999	gasoline	False	gasoline	3.0	sedan	False	owned
4	automatic	black	280000	2001	gasoline	False	gasoline	2.5	universal	False	owned
...	...	...	...	...	...	...	...	...	...	...	...
38526	automatic	silver	290000	2000	gasoline	False	gasoline	3.5	sedan	False	owned
38527	mechanical	blue	321000	2004	diesel	False	diesel	2.2	hatchback	False	owned
38528	automatic	blue	777957	2000	gasoline	False	gasoline	3.5	sedan	False	owned
38529	mechanical	black	20000	2001	gasoline	False	gasoline	2.0	minivan	False	owned
38530	automatic	silver	297729	2000	gasoline	False	gasoline	2.4	minivan	False	owned

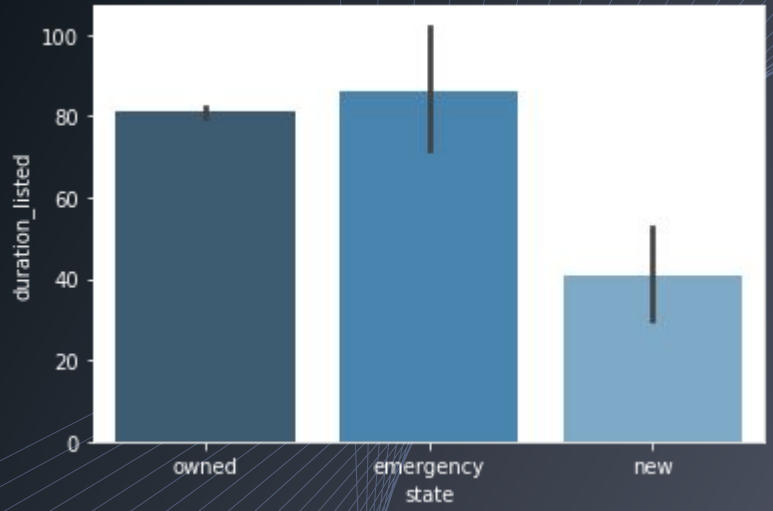
	transmission	color	odometer_value	year_produced	engine_fuel	engine_has_gas	engine_type	engine_capacity	body_type	has_warranty	state	drive
0	0	9	190000	2010	0	0	0	2	11	0	0	
1	0	2	290000	2002	0	0	0	3	11	0	0	
2	0	8	402000	2001	0	0	0	2	10	0	0	
3	1	2	10000	1999	0	0	0	3	9	0	0	
4	0	1	280000	2001	0	0	0	2	11	0	0	

Visualizing the data

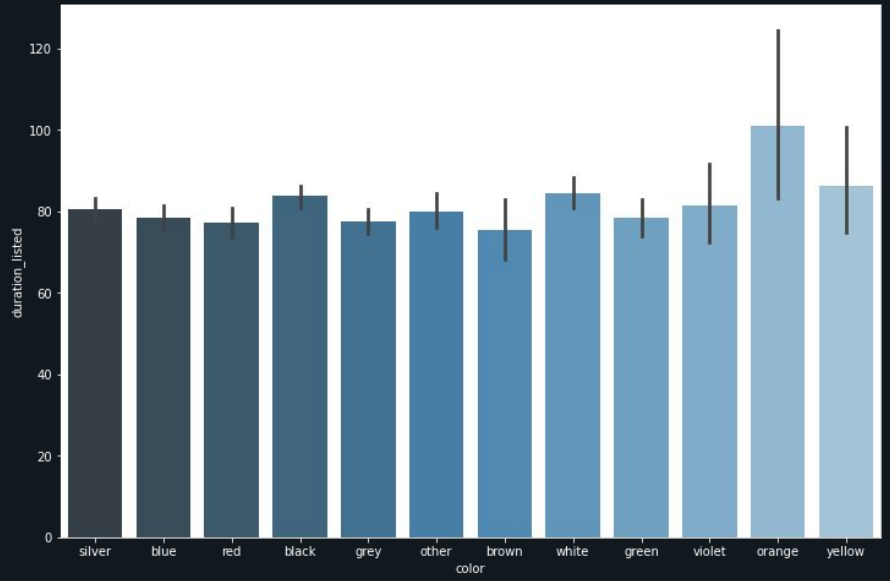
In this process we made use of seaborn and matplotlib to have the data in its graphical form.

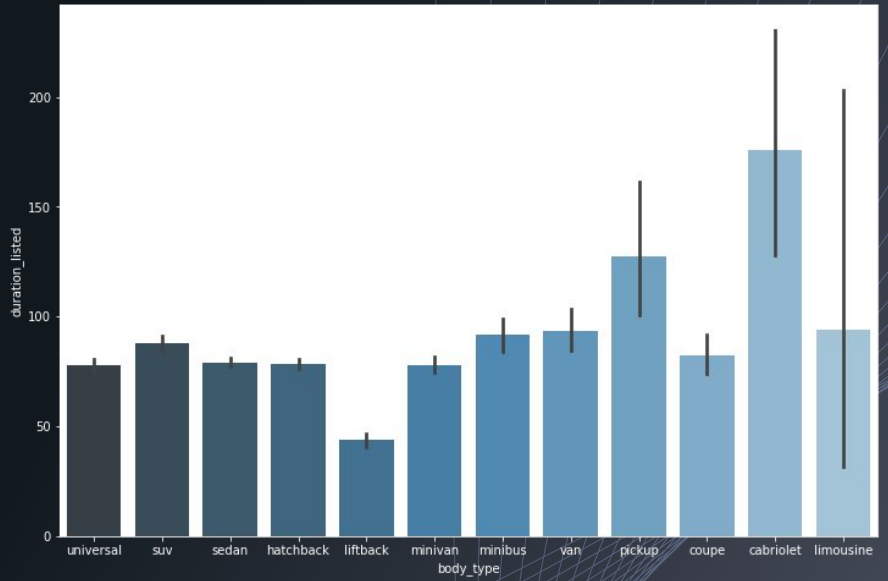


This plot shows how engine fuel influences the duration listed.



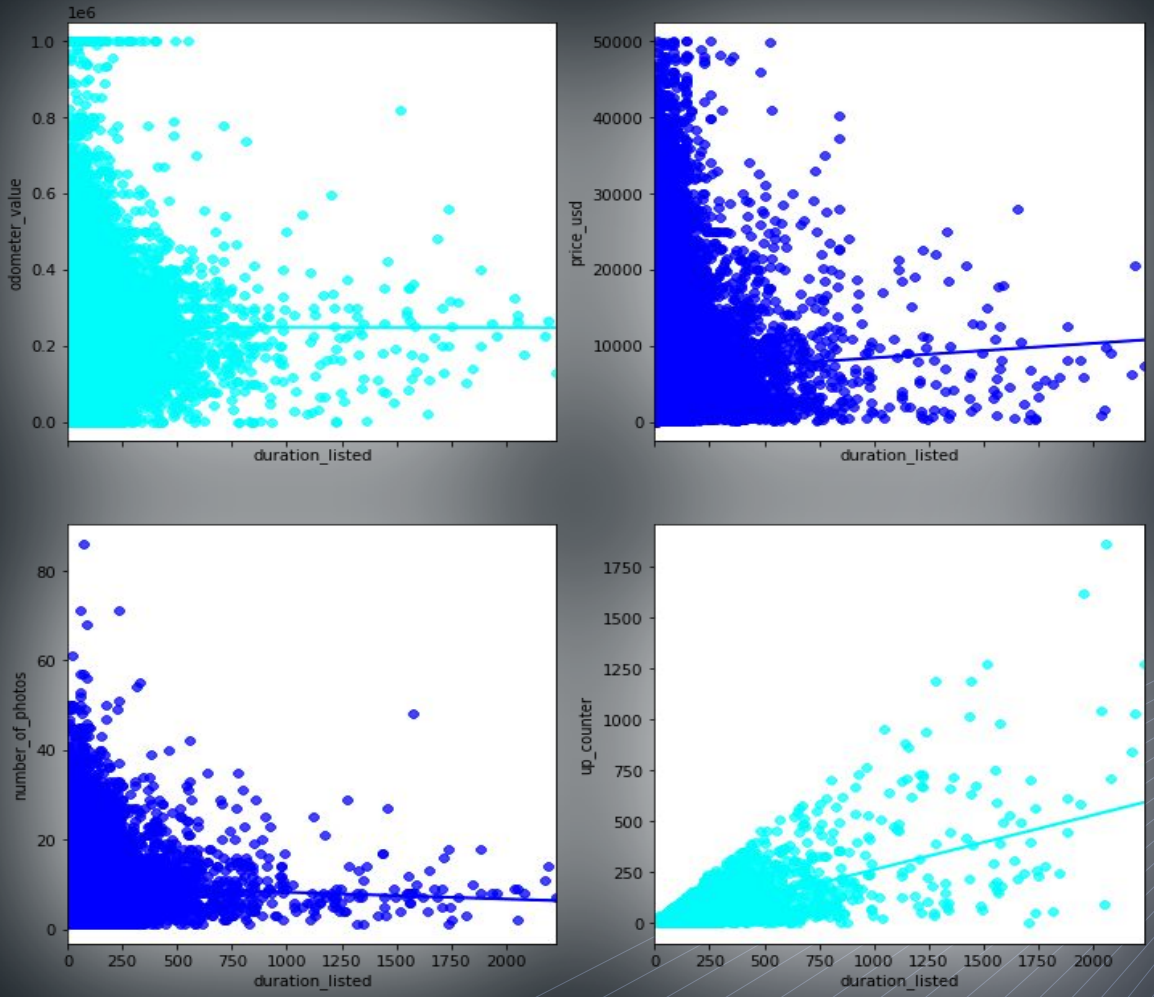
This bar plot shows how the state from where the car belongs influences the duration listed.





This bar plot shows how the color of the car influences the duration listed.

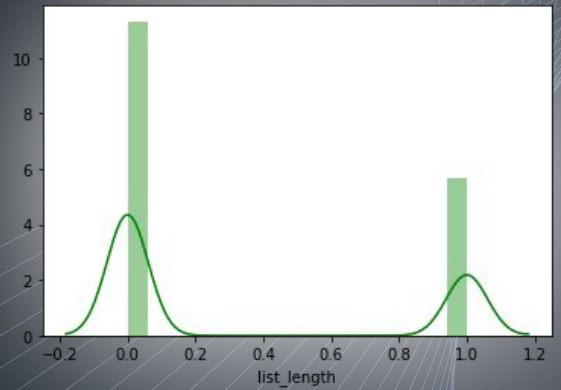
This plot shows how the body type of the car influences the duration listed.



Displays the relationship between the duration listed and odometer_value, price, number of pictures, and up_counter.

Turning the dependent variable into a binary variable

- Initially our dependent variable was a value that showed the number of days that it was listed for sale.
- Created binary variable to split data into two categories long sale postings and short sale posting.
- Split data by whether the sale length was longer (1) or shorter (0) than the mean sale posting



Machine Learning Models

- Created Logistic Regression, KNN, Random Forest, and Decision Tree Models
- Initial models showed accuracy at 100% or close which made us suspect overfitting.
- Removed duration_listed because of redundancy from dataset.

```
LogReg CV (mean, std) 0.9985538250545446 0.002013486577885847

array([[7631,    3],
       [   0, 3923]], dtype=int64)
```

```
DTC CV (mean, std) 1.0 0.0

array([[7634,    0],
       [   0, 3923]], dtype=int64)
```

Machine learning results

- Without overfitting we obtained much more realistic model results
- **KNN:** Accuracy: 0.6139, Std Dev: 0.009
  - Confusion matrix : [6175, 1459], [3018, 905]
- **Logistic Regression:** Accuracy: , 0.736 Std Dev: 0.008
  - Confusion matrix : [7114, 520], [2691, 1232]
- **Decision Tree:** Accuracy: 0.6697, Std Dev: 0.004
  - Confusion matrix : [5698, 1936], [1931, 1992]
- **Random Forest:** Accuracy: 0.7347, Std Dev: 0.006
  - Confusion matrix : [6860, 774], [2398, 1525]
- The Random Forest is our best model even though the accuracy is not super high it is better than a random guess

Random Forest Variable Importance

- Transmission: 0.01279033
- Color: 0.06514133
- **Odometer_value: 0.14053184**
- **Year_produced: 0.09307816**
- Engine_fuel: 0.01424081
- Engine_has_gas: 0.00471782
- Engine_type: 0.01016203
- Engine_capacity: 0.02287148
- Body_type: 0.04875993
- Has_warranty: 0.00133687
- State: 0.00259085
- Drivetrain: 0.01729831
- **Price_usd: 0.13526966**
- Is_exchangeable: 0.01461194
- **Number_of_photos: 0.09480375**
- **Up_counter: 0.3217949**

Linear Regression

Pre Log Transformation

R^2 Train:	0.5037774359874632
R^2 Test:	0.48538065141578823
('transmission', -0.9408400712863816), ('color', -0.041588834467449244), ('odometer_value', -2.3146512433449803e-06), ('year_produced', -0.2954324444368248), ('engine_fuel', -0.4184859805269843), ('engine_has_gas', 5.669278511959426), ('engine_type', -0.22885544488994064), ('engine_capacity', 3.454841932430531), ('body_type', -0.2310277538438681), ('has_warranty', -33.36397881920073), ('state', -5.910081492883939), ('drivetrain', 1.7015454010293172), ('price_usd', 0.0006738125169388676), ('is_exchangeable', -8.98218185306711), ('number_of_photos', -1.4809075698554923), ('up_counter', 1.794374601830004)	MSE train: 79.74247053747429 MSE test: 80.32640132188781

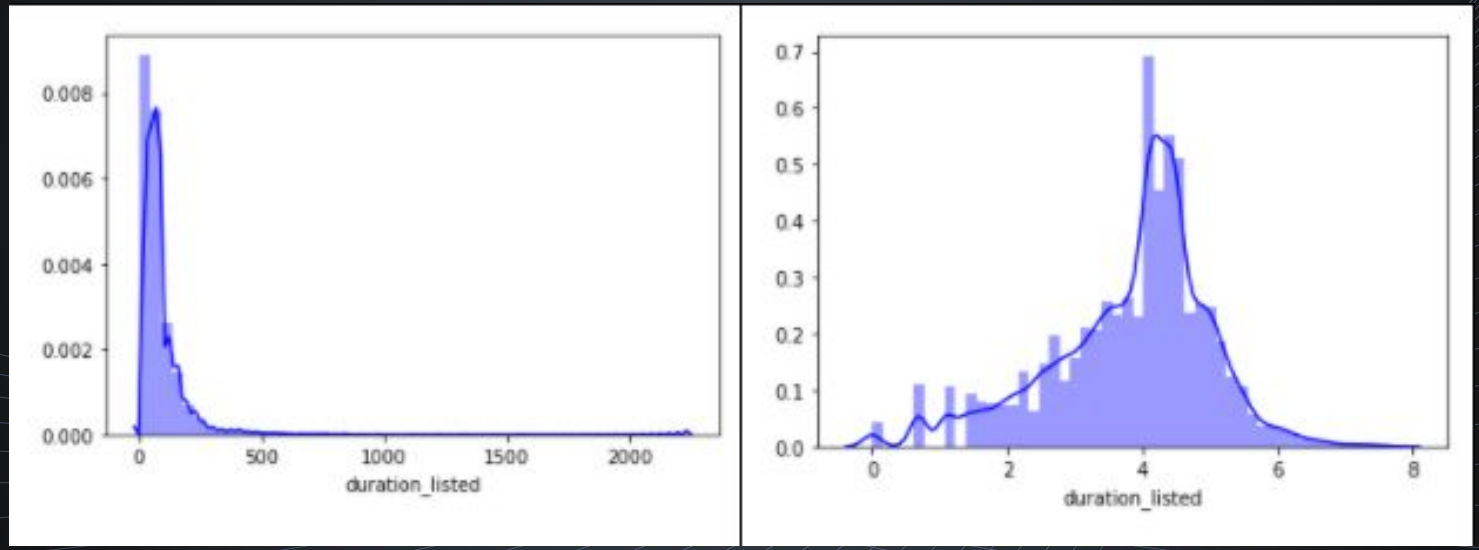
Post Log Transformation

```
duration_listed = (np.log1p(df['duration_listed']))
```

R^2 Train:	0.1690187284152861
R^2 Test:	0.18126759476250864
('transmission', -0.023218381769398302), ('color', 0.003783514893310679), ('odometer_value', 5.024553995828407e-07), ('year_produced', 0.0020509280327624837), ('engine_fuel', -0.02192709075201836), ('engine_has_gas', 0.0733365311519151), ('engine_type', -0.021943637011473765), ('engine_capacity', 0.02741489810083477), ('body_type', -0.0064806217921089026), ('has_warranty', -1.4037545208110935), ('state', -0.18537471899813723), ('drivetrain', -0.007561708186708968), ('price_usd', 1.578774715273052e-05), ('is_exchangeable', -0.06816435360782955), ('number_of_photos', -0.023606039350039205), ('up_counter', 0.009581369098249472)	MSE train: 1.0839454601214498 MSE test: 1.0736843478830824

Linear Regression models before and after Log transformation.

Distribution of our dependent variable, duration listed, before and after log transformation.



Take away from Linear Regression

Running the linear regression, we got a moderately high R^2 value on the initial model. The value is nearly 0.5 which tells us that our model fits about 50% of our data, and thus can be used moderately accurate predicting.

Although this linear regression provided some insight on the important variables, there was still significant error. This is why we decided to use a log transformation on the dependent variable, list_duration. This helped normalize the dependent variable distribution. After running the log transformation we ran the linear regression model again. However, the R^2 value decreased drastically leading us to prefer our original model.

Ridge and Lasso Model

- To improve the predictiveness of our linear regression model, we executed Ridge and Lasso regularization models on the data.
- These models penalize the coefficients in a linear model that get too large.
- Large coefficients can lead to overfitting on the training data and thus decrease the predictability of a model on unseen data.
- The difference between the ridge and lasso models is that the ridge model keeps all variables in the model after regularization, while the lasso model completely removes certain variables from the model

Ridge Model:	
R Squared:	0.48537827323971905
MSE:	80.3265869251936

('transmission', -0.9425607790660513),
('color', -0.04156816454779607),
('odometer_value', -2.3088536517230834e-06),
('year_produced', -0.29556535064515227),
('engine_fuel', -0.4085276209174005),
('engine_has_gas', 5.64627446186773),
('engine_type', -0.23779902006584566),
('engine_capacity', 3.4576167950632217),
('body_type', -0.2313055395603744),
('has_warranty', -32.866325894550926),
('state', -6.06790002143534),
('drivetrain', 1.6997589755995388),
('price_usd', 0.0006728062074749671),
('is_exchangeable', -8.98502670470948),
('number_of_photos', -1.4810327767419336),
('up_counter', 1.7944143558496197)

Lasso Model:	
R Squared:	80.60408864293389
MSE:	0.4818164365803722

('transmission', -0.0),
('color', -0.0),
('odometer_value', 3.020815481537689e-06),
('year_produced', -0.2783185138155344),
('engine_fuel', -0.0),
('engine_has_gas', 0.0),
('engine_type', -0.0),
('engine_capacity', 0.03279027318713382),
('body_type', -0.0),
('has_warranty', -0.0),
('state', -0.0),
('drivetrain', 0.0),
('price_usd', 0.0005920363487138592),
('is_exchangeable', -0.8617932398343402),
('number_of_photos', -1.482867613275572),
('up_counter', 1.7914059005173764)

Conclusion

- Certain features have high influence
  - Engine_capacity: 3.45
  - Has_warranty: -33.36
  - State: -5.91
  - Is_exchangeable: -8.98
  - Number_of_photos: -1.48
  - Up_counter: 1.79

- It is important for a business to focus on the listing as much, if not more, than the actual car they are selling.

Packages used

- Pandas to read the data frame
- seaborn to visualize graphs
- numpy to work with arrays
- matplotlib.pyplot to plot graphs
- sklearn to use machine learning
- sklearn.metrics to measure classification performance (accuracy and confusion matrix)
- sklearn.discriminant_analysis to classify observations
- sklearn.neighbors to work on KNN
- sklearn.tree to use decision trees
- sklearn.naive_bayes used as classification algorithms
- sklearn.svm tp support vector machines
- sklearn.inspection to use tools for model inspection
- from sklearn.linear_model import LinearRegression

Thanks!

Any questions?