Used Car Analysis

BUDT 737: Big Data and Artificial Intelligence for Business

Group 1: 3.30pm

Shubham Prakash, Shweta Dudhal, Atharva Patil

Business Problem

Used Cars Dealers face difficulty in understanding how to buy/refurbish, coming up with a pricing strategy for the cars and the location of their stores.

Business Questions

1. What type of used cars (car models) / Brand name cars are sold the most?
2. Which cities have the highest average car prices?
3. Does a car belonging to a particular used category (car history) affect the price of the car?
4. What colour (interior colour/exterior colour) cars are sold the most?
5. Does the value of a car fall over the years?

Business Impact

The predictive model will help the dealer estimate the correct price for a car. It will also help the dealer understand what factors influence the price of the end to end.

Data Source and Description

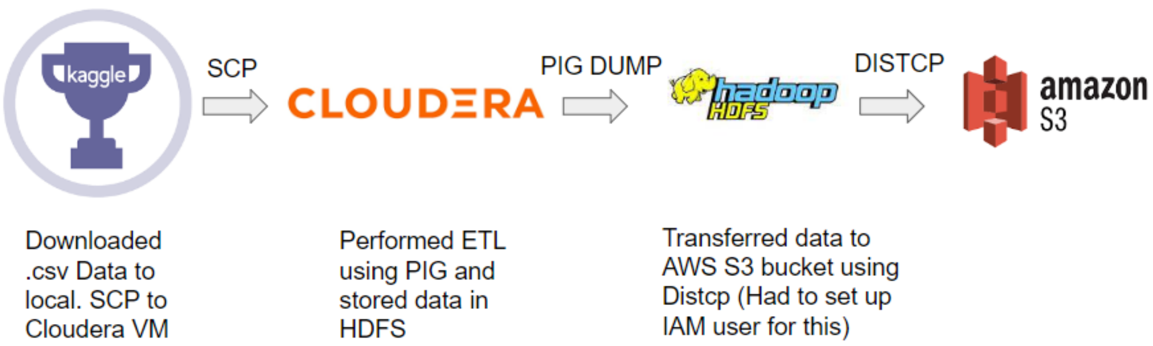
We downloaded the data from Kaggle (Link).

Each row represents the sale of a used car. There are 66 variables describing price, car maker, car type, engine type, exterior color, build year, city, etc. Total size of the data is 9.8GB with 3M rows.

We were facing the issue of 'Disk Space Full' while copying this data to Cloudera due to the sheer size of the data (The Cloudera VM from class had an available storage of only 8GB). Hence, we had to sample the data to 2GB. We did this using the csv splitting software- 'CSV Splitter' from the website.

Data Architecture

We had the following data architecture for the ETL process:



We downloaded the data (.csv file) from Kaggle, sampled it (as explained earlier) and then copied it to the Cloudera VM

On the Cloudera VM, we ran a PIG script, which extracted the data from the local file storage in the Cloudera VM, transformed it (Defined a metadata for the data and removed the first row, which was all headers) and loaded it into the Cloudera Hadoop file distributed system.

From the Cloudera HDFS we transferred the data to an Amazon S3 bucket using the distcp command. We chose to store the data on AWS S3 bucket because it makes our data universally accessible. We can now load our data on different platforms like Databricks, Google Colab, Tableau, etc. This gives us more flexibility while performing data analysis as well as predictive modelling.

Below is the PIG ETL script that we used:

```
input_file = LOAD '/used_cars_sample.csv' USING PigStorage (',') AS (vin:chararray, back_legroom:chararray, bed:chararray, bed_height:chararray, bed_length:
daysonmarket:int, dealer_zip:chararray, description:chararray, engine_cylinders:chararray, engine_displacement:float, engine_type:chararray,
exterior_color:chararray, fleet:chararray, frame_damaged:chararray, franchise_dealer:chararray, franchise_make:chararray,
front_legroom:chararray, fuel_tank_volume:chararray, fuel_type:chararray, has_accidents:chararray, height:chararray,
highway_fuel_economy:float, horsepower:float, interior_color:chararray, isCab:chararray, is_certified:chararray, is_cpo:chararray,
is_new:chararray, is_oemcpo:chararray, latitude:double, length:chararray, listed_date:chararray, listing_color:chararray,
listing_id:chararray, longitude:double, main_picture_url:chararray, major_options:chararray, make_name:chararray,
maximum_seating:chararray, mileage:float, model_name:chararray, owner_count:float, power:chararray, price:float, salvage:chararray,
savings_amount:int, seller_rating:double, sp_id:chararray, sp_name:chararray, theft_title:chararray, torque:chararray, transmission:chararray,
transmission_display:chararray, trimId:chararray, trim_name:chararray, vehicle_damage_category:chararray, wheel_system:chararray,
wheel_system_display:chararray, wheelbase:chararray, width:chararray, year:chararray);

ranked = rank input_file;

noheader = Filter ranked by (rank_input_file>1);

new_input_file = foreach noheader generate vin, back_legroom, body_type, city, city_fuel_economy, daysonmarket,
dealer_zip, engine_cylinders, engine_displacement, engine_type, exterior_color, fleet, frame_damaged, franchise_dealer,
franchise_make, front_legroom, fuel_tank_volume, fuel_type, has_accidents, height, highway_fuel_economy, horsepower,
interior_color, isCab, is_new, latitude, length, listed_date, listing_color, listing_id, longitude, major_options,
make_name, maximum_seating, mileage, model_name, owner_count, power, price, salvage, savings_amount, seller_rating,
sp_id, sp_name, theft_title, torque, transmission, transmission_display, trimId, trim_name, wheel_system, wheel_system_display, wheelbase, width, year;

STORE new_input_file INTO '/big_data_project/' USING PigStorage (',');
```

Here is how the loaded data in the Cloudera HDFS looks like:

```
[training@ip-172-31-82-187 ~]$ hdfs dfs -ls /big_data_project/
Found 5 items
-rw-rw-rw-   1 training supergroup          0 2022-12-05 10:19 /big_data_project/_SUCCESS
-rw-rw-rw-   1 training supergroup  126888945 2022-12-05 10:18 /big_data_project/part-m-00000
-rw-rw-rw-   1 training supergroup  122349476 2022-12-05 10:18 /big_data_project/part-m-00001
-rw-rw-rw-   1 training supergroup  117582805 2022-12-05 10:18 /big_data_project/part-m-00002
-rw-rw-rw-   1 training supergroup  112104167 2022-12-05 10:19 /big_data_project/part-m-00003
```

Below is the command we used to transfer data from hdfs to AWS S3**:

hadoop distcp -Dfs.s3a.access.key="AKIA3XQXYXODX2ZCKVG5" -Dfs.s3a.secret.key="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" /big_data_project/part s3a://group1bigdatabucket/data/*

** Before running this command, we had to set up an IAM user on S3 to the required credentials (fs.s3a.access.key, fs.s3a.secret.key)

Here is how the AWS S3 bucket looks like after data transfer:

Amazon S3 > Buckets > group1bigdatabucket > data/

data/

Copy S3 URI

Objects | Properties

Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix | Show versions | 1

Name	Type	Last modified	Size	Storage class
part-m-00000	-	December 6, 2022, 05:07:12 (UTC-05:00)	121.0 MB	Standard
part-m-00001	-	December 6, 2022, 05:07:12 (UTC-05:00)	116.7 MB	Standard
part-m-00002	-	December 6, 2022, 05:07:12 (UTC-05:00)	112.1 MB	Standard
part-m-00003	-	December 6, 2022, 05:07:12 (UTC-05:00)	106.9 MB	Standard

VM Specifications

We used Cloudera VM for the ETL process. It is the same VM that we used throughout our classwork.

Below are the specifications of the AMI (ami-060b9e8d0cfea279f) it is using:

AMI ID ami-060b9e8d0cfea279f	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name Cloudera Virtual Machine	Owner account ID 949991318335	Architecture x86_64	Usage operation RunInstances
Root device name /dev/sda1	Status Available	Source 949991318335/Cloudera Virtual Machine	Virtualization type hvm

Below are VM specifications (Instance Type and Storage):

Instance ID i-0c4f6abf798b05e32 (Cloudera)	Public IPv4 address 54.166.189.120 | open address	Private IPv4 addresses 172.31.82.187
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-54-166-189-120.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-82-187.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-82-187.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –

Volume ID	Size	Type	Volume status
vol-06248e5135d0639bf	20 GiB	gp2	Okay
Volume state	IOPS	Throughput	Encryption
In-use	100	-	Not encrypted
KMS key ID	KMS key alias	KMS key ARN	Snapshot
-	-	-	snap-00de6ac57ba7e94e0
Availability Zone	Created	Multi-Attach enabled	Attached Instances
us-east-1c	Thu Sep 08 2022 16:02:34 GMT-0400 (Eastern Daylight Time)	No	i-0c4f6abf798b05e32 (Cloudera): /dev/sda1 (attached)
Outposts ARN			
-			

Data and Predictive Analysis Architecture

Below is the architecture that we used:

After we loaded the data on AWS S3, we started a Google Colab notebook with a Google hosted runtime and connected our notebook to the S3 bucket using the BOTO3 function in python to fetch the data as a pandas dataframe.

Below is the python script using BOTO3 to load the data from AWS S3 to Google Colab notebook-

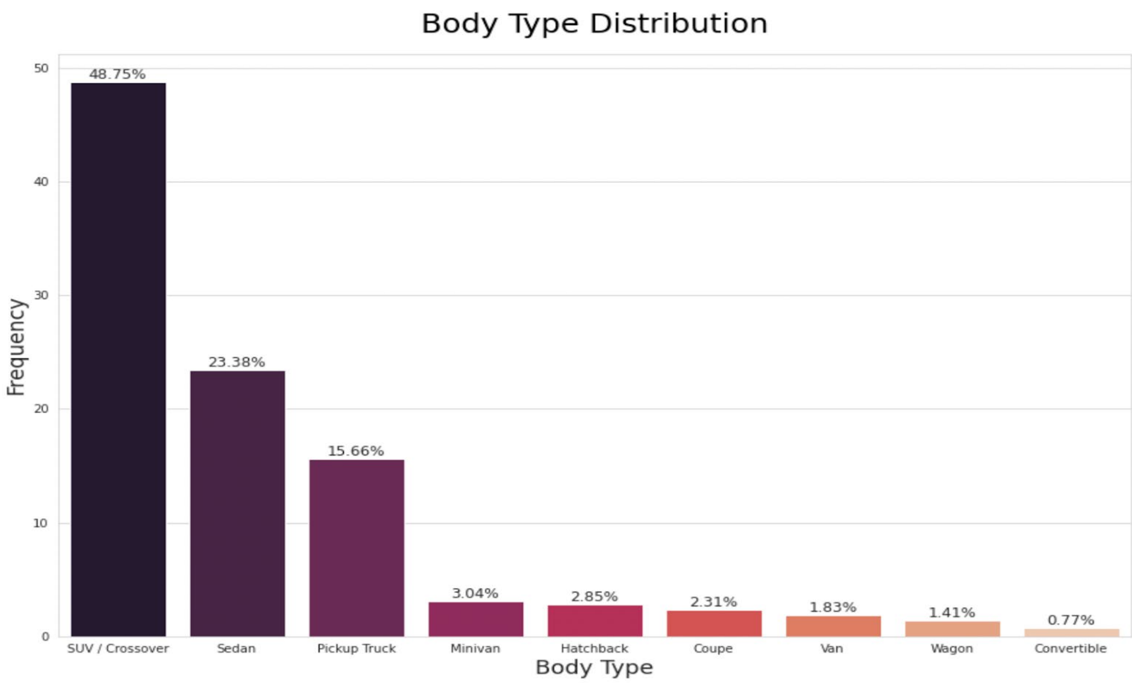
```python
s3 = boto3.resource(
    service_name='s3',
    region_name='us-east-1',
    aws_access_key_id='AKIA3XQXYXODX2ZCKVG5',
    aws_secret_access_key='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'
)

for bucket in s3.buckets.all():
    print(bucket.name)

for obj in s3.Bucket('group1bigdatabucket').objects.all():
    print(obj)

bucket_list=[]
for file in s3.Bucket('group1bigdatabucket').objects.all():
  file_name=file.key
  if file_name=='data/':
    continue
  else:
    bucket_list.append(file_name)

df=[]
for part in bucket_list:
  obj=s3.Object('group1bigdatabucket',part)
  data=obj.get()['Body'].read()
  df.append(pd.read_csv(io.BytesIO(data),header=None))

cars=pd.DataFrame(columns=['vin','back_legroom','body_type','city','city_fuel_economy','daysonmarket','dealer_zip','engine_cylinders','engine_displacement',
                           'engine_type','exterior_color','fleet','frame_damaged','franchise_dealer','franchise_make','front_legroom','fuel_tank_volume',
                           'fuel_type','has_accidents','height','highway_fuel_economy','horsepower','interior_color','isCab','is_new','latitude','length',
                           'listed_date','listing_color','listing_id','longitude','major_options','make_name','maximum_seating','mileage','model_name',
                           'owner_count','power','price','salvage','savings_amount','seller_rating','sp_id','sp_name','theft_title','torque','transmission',
                           'transmission_display','trimId','trim_name','wheel_system','wheel_system_display','wheelbase','width','year'])

for data in df:
  cars_data=pd.DataFrame(data=data)
  cars=pd.DataFrame(np.concatenate([cars.values, cars_data.values]),columns=cars.columns)
```

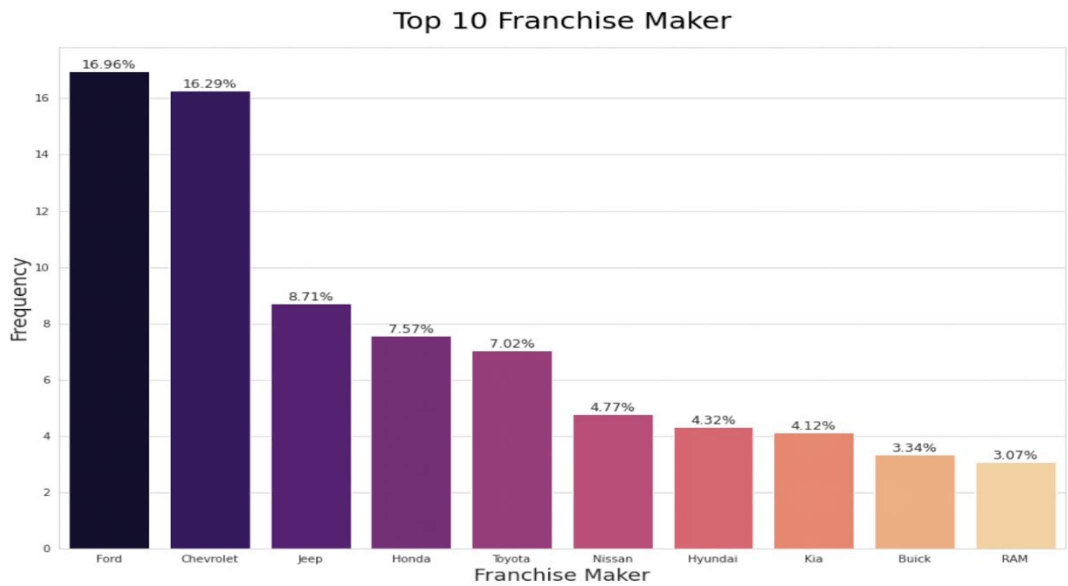
Once the data was loaded to Google Colab, we performed the Descriptive analysis using python whereas the predictive analysis using PySpark.

Descriptive Analysis with Python

1. **What type of used cars (car models) / Brand name cars are sold the most?**

   Variable Used: body_type, franchise_make

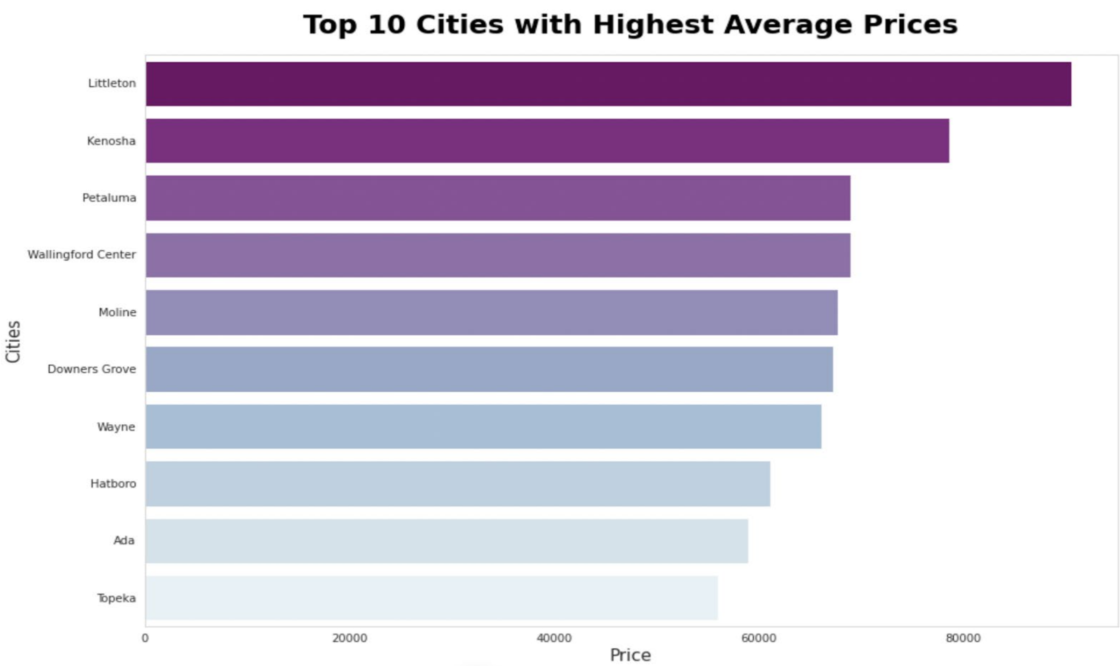




We can see that the top three model types are SUV/Crossover, Sedan and Pickup trucks. Additionally, the top leading brands are Ford, Chevrolet and Jeep. Thus, investing in these models and these top brands will be beneficial for the dealers as the probability of the car being sold will be higher. Additionally, they will be sold faster as compared to other cars belonging to different models and brands.

2. **Which cities have the highest average car prices?**

Variable Used: city, price
Grouped the prices as per the city and took average of the price. Filtered only the top 10 cities having highest average price

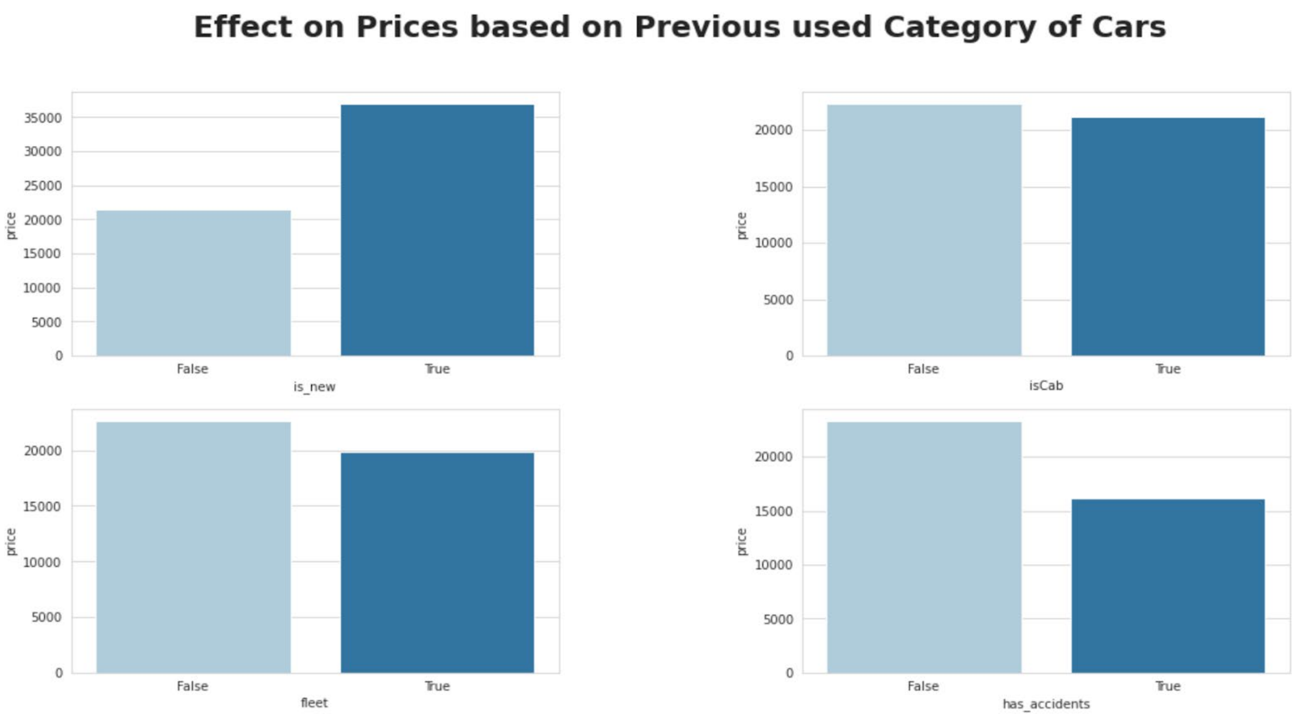


Littleton is the city with the highest average price. Knowing this information will help the dealers belonging to a particular city come up with their pricing strategy. If a dealer wants to sell a car in Littleton he has to make sure that the car is equal to or below the average selling price of cars in the city. This will ensure that the buyer buys the car from them and not their competitor. It will increase the dealer sale and revenue. If the car is priced higher than the average car price in that city, it has less chances of being sold.

3. **Does a car belonging to a particular used category (car history) affect the price of the car?**

   Variable Used: is_new, isCab, fleet, has_accidents (categorical variables), price

   Calculated average price of car when it was a cab vs when it was not a cab. Performed the same computation on every other categorical variable.

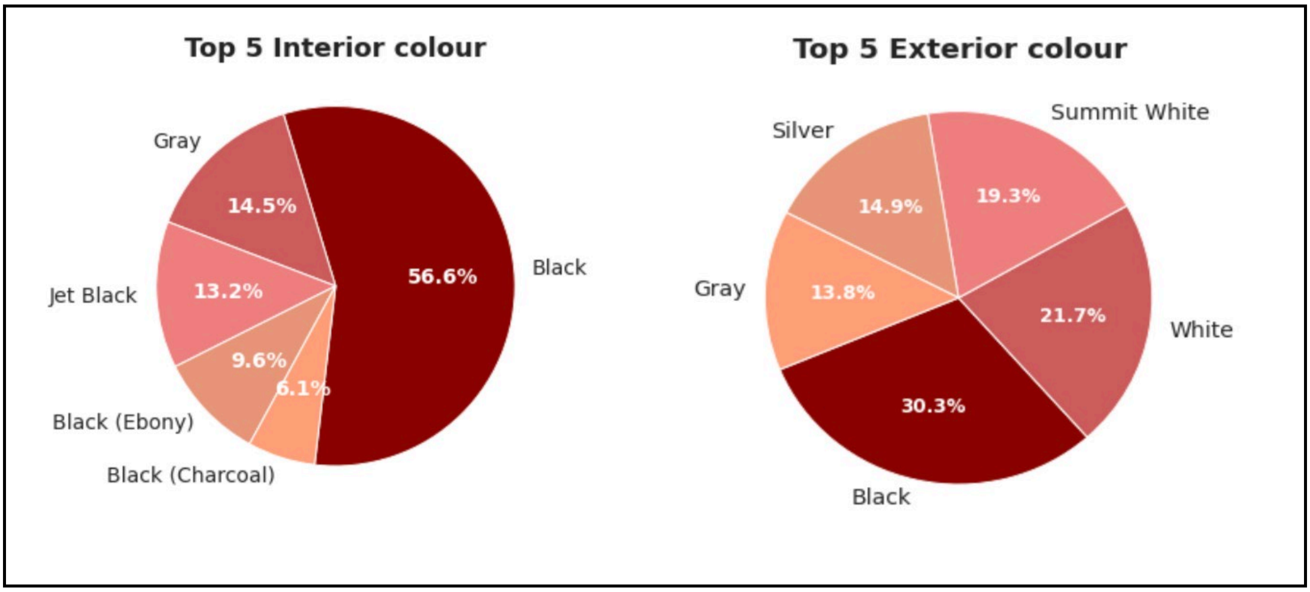


   This graph would help the dealer decide which category used car he should invest in. The highest average price is for a new car, i.e, a car which is used for less than 2 yr. The price of this car would be high because it is relatively new and well maintained. However, the cars that were previously used as cabs or were part of a fleet or have been in accidents are low priced. Mainly, because they were not maintained properly and are damaged. The lowest average price is of the car that has been in accidents. Therefore, investing in cars that have been in accidents previously would not be a good idea.

4. **What color (interior color/exterior color) cars are sold the most?**

Variable Used: interior_color, exterior_color

Filtered top 5 colours for both categories



Black is the most preferred colour. So, Investing in a black car would be better as it is a safe colour and likely to sell faster in future.

5. Does the value of a car fall over the years?

Variable Used: year (built year: the year car was manufactured), price
Calculated average price of car over past 52 years

This graph will help the dealer decide the price of a car based on the year the car was manufactured. It is evident that the prices of cars have been increasing since 2000. It is mainly because the cars manufactured during this period and post that are modernized, have more features and are highly equipped with technology. Also, when we look at the prices of the cars built in the 1990's or in the past, they are highly priced as they are vintage and are highly valuable. Cars that were manufactured way in the past are low priced, as they are old and would not be functional now. Additionally, it would also help the dealer decide which car they should invest in. Thus, it would be ideal for the dealer to buy recently manufactured cars as well the cars that are considered vintage now and are highly valuable.

Predictive Modelling with Pyspark

- **Initializing Spark Session and loading data in an RDD Dataframe**

We first initialized a spark session and then added our AWS credentials to establish connection with AWS

```
import os
import sys

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, Row
SPARK_CONF = SparkConf().setAppName("used_car_project")
spark = SparkSession.builder    .config(conf=SPARK_CONF)     .enableHiveSupport().getOrCreate()
print("Spark session initialized")
spark._jsc.hadoopConfiguration().set('Dfs.s3a.aws.credentials.provider','org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider')
spark._jsc.hadoopConfiguration().set('fs.s3a.access.key','AKIA3XQXYXODX2ZCKVG5')
spark._jsc.hadoopConfiguration().set('fs.s3a.secret.key','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')
```

Then we loaded the used cars data in a spark dataframe (RDD) from the previously extracted data from S3 (This was done using Boto3 and data was loaded in a pandas dataframe, as discussed earlier)

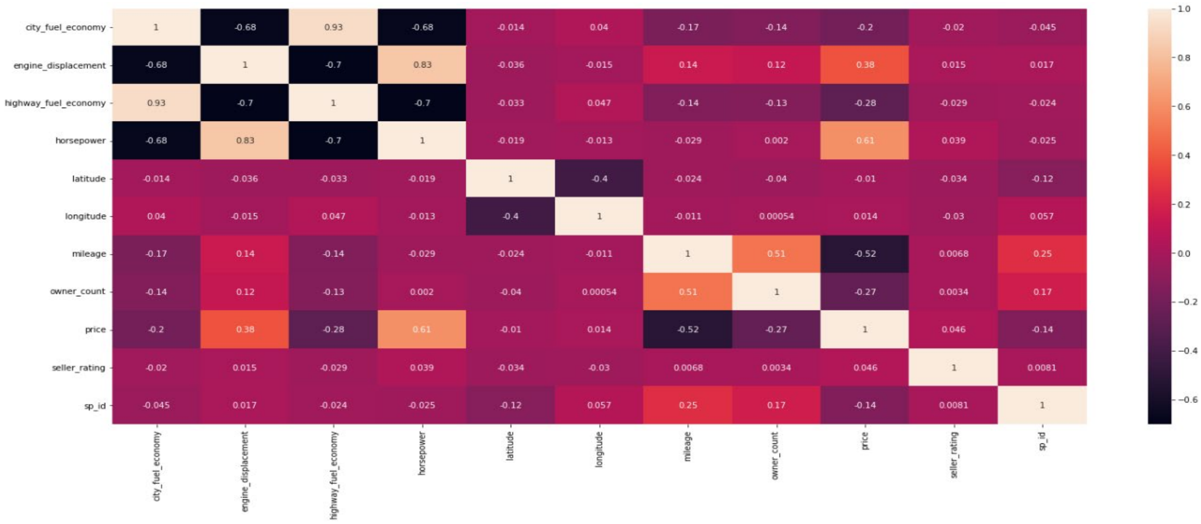
```
[129] cars_new=cars[['body_type', 'city', 'city_fuel_economy',
              'daysonmarket', 'engine_cylinders',
              'engine_displacement', 'engine_type', 'exterior_color', 'fleet',
              'frame_damaged', 'franchise_dealer', 'franchise_make', 'front_legroom',
              'fuel_tank_volume', 'fuel_type', 'has_accidents', 'height',
              'highway_fuel_economy', 'horsepower', 'interior_color', 'isCab',
              'is_new', 'length', 'listed_date', 'listing_color',
              'make_name', 'maximum_seating', 'mileage', 'model_name', 'owner_count',
              'power', 'price', 'salvage', 'savings_amount', 'seller_rating',
              'theft_title', 'transmission',
              'transmission_display', 'wheel_system',
              'wheel_system_display', 'wheelbase', 'width', 'year']]
```

```
[130] cars_new['fleet'].fillna(False,inplace=True)
      cars_new['frame_damaged'].fillna(False,inplace=True)
      cars_new['has_accidents'].fillna(False,inplace=True)
      cars_new['isCab'].fillna(False,inplace=True)
      cars_new['salvage'].fillna(False,inplace=True)
      cars_new['theft_title'].fillna(False,inplace=True)
      cars_new = cars_new.apply(lambda x: x.fillna(0) if x.dtype.kind in 'biufc' else x.fillna('unknown'))
      cars_sparkdf = spark.createDataFrame(cars_new)
```

- **Building Correlation Matrix**

In the dataset to figure out which features factor into the determination of the price of these used cars, we plotted a correlation matrix.

The heatmap proves to be an efficient tool to visualize the impact of these features on the targeted variable i.e. price.



According to the correlation matrix above we can see that certain features impact the price much more than others.

Engine displacement and horsepower have the most positive correlation whereas highway fuel economy, mileage and owner count have the most negative correlation with price. It makes sense since price should increase with better engine displace and higher horsepower whereas decrease as the number of owners increase. What we were surprised about was that the price was negatively correlated to mileage. Ideally, we expected mileage to be positively correlated to price.

- **Applying Regression Model**

Now that we have a better idea of which features might help us making better price predictions, we decided to go ahead with the Random Forest regression model using the features we just discussed. For this, we needed to convert the feature variables into a feature vector before passing to the model.

We did the following for feature vector extraction:

- We used ‘StringIndexer to one hot encode the categorical variables and then imputed them into a vector.

- Then, we imputed the missing numerical variables to the median and imputed them into a vector
- We created a final feature vector by imputing the categorical and numerical vectors together using 'VectorAssembler'

Below is the PySpark code we used for feature vector extraction:

```
pipe_stages= []

# all string(categorical) variables will be encoded into numbers, each category by frequency of label
sindexer= StringIndexer(inputCols= to_encode,
                        outputCols= ["indexed_{}".format(item) for item in to_encode],
                        handleInvalid='keep',
                        stringOrderType='frequencyDesc')
pipe_stages += [sindexer] # must add each step to the Pipeline

# # dummy numerized strings into a sparse vector. (I didn't need this step, so I left it out)
# ohe= OneHotEncoder(inputCols=["indexed_{}".format(item) for item in to_encode],
#                    outputCols= ["indexed_ohe_{}".format(item) for item in to_encode],
#                    handleInvalid='keep',
#                    dropLast=True)
# pipe_stages += [ohe]

# impute missing numerical values, with the median (though bad practice)
imp= Imputer(inputCols= numerical_cols,
             outputCols=['imputed_{}'.format(item) for item in numerical_cols],
             strategy= 'median')
pipe_stages += [imp]

# create the un-standardized features vector
assembler= VectorAssembler(inputCols= ["indexed_{}".format(item) for item in to_encode] + ['imputed_{}'.format(item) for item in numerical_cols],
                           outputCol= "feats",
                           handleInvalid="keep")
pipe_stages += [assembler]

# scale all features. Maybe you want to do this Before encoding the string columns?
ss= StandardScaler(inputCol="feats",
                   outputCol="features",
                   withMean= False,
                   withStd=True)
pipe_stages += [ss]

pipe= Pipeline(stages= pipe_stages)
```

Once we had our feature vector, we split the data into an 80:20 ratio for training and testing respectively, i.e, 80% of the data was to be used for the training the model whereas 20% of the data was to be used for testing or validating the model.

Then we trained a Random Forest Regressor model (with number of trees=200 and max depth=4) with the training data (containing the feature vector) we just created and scored/predicted it on the testing dataset.

Below is the code for the same-

```
train, test= df.randomSplit([0.80, 0.20], seed=42)
# train size is 80% and test size is 20% of the data
```

```
rfc= RandomForestRegressor(numTrees=200,
                            maxDepth=4,
                            labelCol='price',
                            seed=42)
```

```
rfc.setFeaturesCol("features")
```

```
RandomForestRegressor_81ad875d96f0
```

```
rfc_model= rfc.fit(train)
```

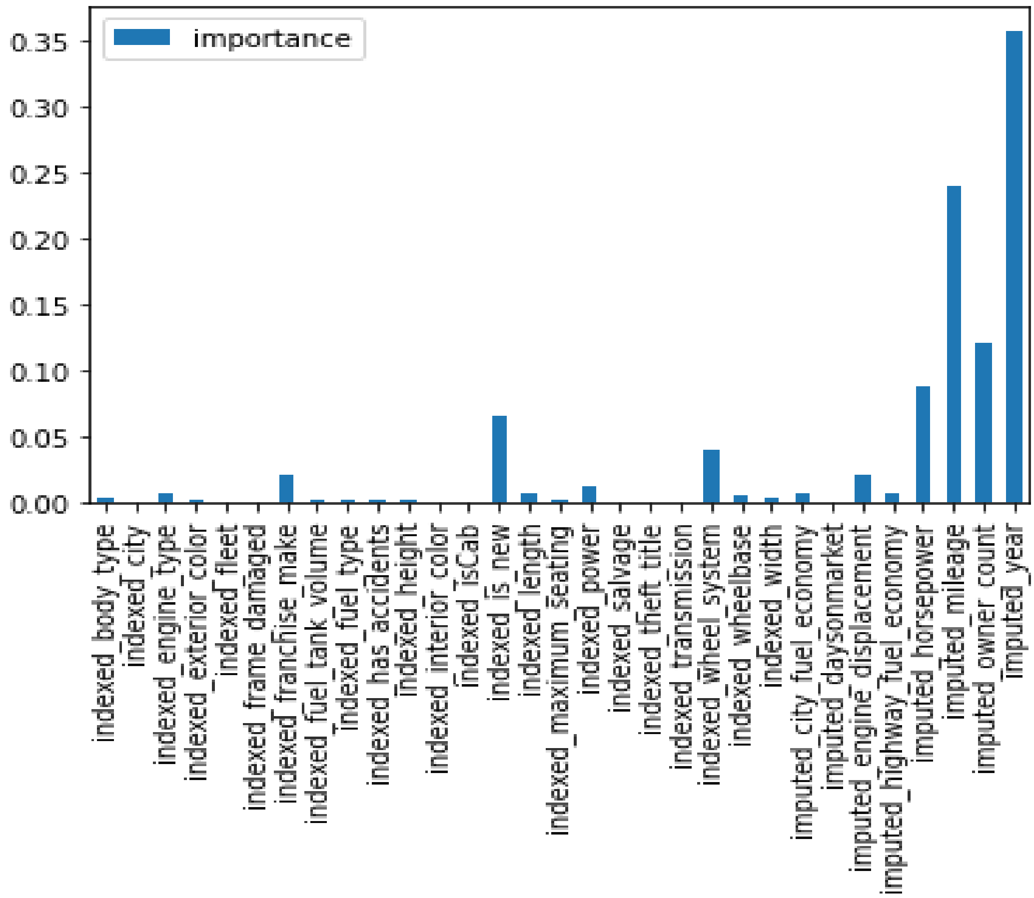
```
rfpredictions = rfc_model.transform(test)
```

The Mean Absolute Error (MAE) for the testing dataset came out to be $451.531, which is almost 10x better than the base model where we just use mean of price of the training dataset as the predicted price for each row of the testing dataset. Thus, our model can be used for predicting price of a used car given its specification.

```
rfevaluator = RegressionEvaluator(predictionCol="prediction", labelCol="price", metricName="mae")
print('MAE:', rfevaluator.evaluate(rfpredictions))
```

```
MAE: 451.5314906637177
```

- **Most Important Features as per the trained Random Forest model**



The feature importance graph from the Random forest model suggests that Build Year (Or Manufacturing year), mileage, Owner Count and car history (is_new flag) come out to be the most important features while predicting the price of a car. This is on similar lines as to what we saw from the descriptive analytics and correlation plot.

Conclusion

A used car dealer needs to keep in mind the following things before making any business decisions:

- Internal Features (especially mileage and horsepower) greatly influence the price of the used cars
- Investing in models like SUV, Sedan and from brands like Ford, Chevrolet will help to sell a vehicle faster
- If a car is fairly new (less than 2yr), it likely to be sold faster. Consequently, a car, which is vintage, is going to be valuable and highly priced.
- History of a car(Like accidents, fleets, number of owners) plays an important role in the price of it
- A predictive model, just like ours, can help them greatly improve their pricing strategy.