CarDealz Test Case Documentation

Introduction

This test case document will be broken into 4 sub divisions accordingly:

1) Data Mining Test Cases - Scraping
2) LR Test Cases - Machine Learning
3) Android Application Test Cases - UI
4) LR Feature Graphical Representation

All testing presented in this document were designed and implemented by David O'Regan.
Of the 100 Test Cases Performed, the below 15-20 have been selected as the most important for user consumption.
Some of the below test cases are a combination of multiple test cases for a summarisation of a testing area.
Described are Unit Testing, Sanity Testing and Smoke Testing.
For a overview of Unit Testing(Spider Contracts, Pytest and Junit) please refer to the project repository: Folders labeled Testing.

<Client> Test Script (*front sheet*)			
Project ID	CarDealz		
AUT Name	CarDealz	**Version**	3.2
Iteration ID	1	**Date of Test**	December - May 13/14

Test ID	Scaper Test 1
Purpose of Test	This test was conducted to confirm that the scraper was successfully harvesting data from www.carzone.ie and that data being harvested was correct i.e. Values were the same as on the webpage. This test was to confirm that the Xpath variables within the scraper were defined properly within context of the data that was to be harvested. After this is verified via a test print out, to see that there is actual values being taken, the next step is validation that the data taken is in fact the same data displayed on the website. This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the dat scraper i.e. Spider. For this test case, the database will not be active and the values will not store, rather the vales being scraped will be displayed via a front end test print for verification manually. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.

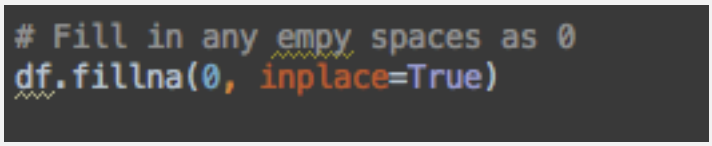
Test Steps	1) Select amount of adds to scrape(10) and notes values of fields being scraped. 2) Point scraper to these add's via URL start points. 3) Allow scraper to run 4) Print scraped data to screen 5) Manually verify each field for correctness
Expected Result	For each of the add's being tested, it is expected that the values the scraping program displays are exactly the same as those on the website advert. Example:  OutPut: Title: Audi A4, Price: 38,995, EngineSize: 2.0, EngineType: Diesel, Transmission: Manual, Year: 2014, Colour: Monsoon Grey, Mileage: 3,106, Owners: 1, Location: Kildare, NCT: " "

Test ID	Scraper Test 2
Purpose of Test	This test was conducted to confirm that the scraper was successfully harvesting data from www.carzone.ie and that data being harvested is stored correctly in the Database table i.e. Values were the same as on the webpage. This test was to confirm that the the values being taken from the www.carzone.ie are not only being scraped correctly, but being stored into the DB table correctly. This relies on a ETL process(Extraction, transformation and loading). This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the dat scraper i.e. Spider. For this test case, the database will be active and the values will be stored. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.

Test Steps	1) Select amount of adds to scrape(10) and note values of fields being scraped. 2) Point scraper to these add's via URL start points. 3) Allow scraper to run 4) Print scraped data to screen & store data in our created DB table 5) Manually verify that data was stored in our table 6) Manually verify that data stored in our DB was correct when compared against original add
Expected Result	For each of the add's being tested, it is expected that the values the scraping program displays are exactly the same as those on the website advert.

title	link	price	carYear	location	mileage	engineSize	engineType	Transm
Audi A4 2.0 TDI 150 Quattro S Line - Audi North Du...	http://www.carzone.ie/search/Audi/A4/2.0-TDI-/3871...	46,995	2014	Dublin	2,485	2	Diesel	Manual

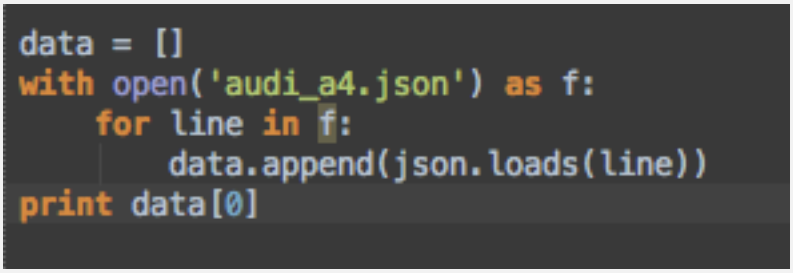
Test ID	Scraper Test 3
Purpose of Test	This test was a unit test that was carried out via spider contracts(Scrapy provided Unit Testing suite) to ensure that the the URL being parsed is in fact correct and stable: http://doc.scrapy.org/en/latest/topics/contracts.html The test is designed to ensure that when the spider is parsing an advert page, it can return the sub URL to enter for deep data harvesting.
Test Environment	
Test Steps	
Expected Result	For each of the add's being tested, it is expected that the spider contract will return nothing on a valid sub-URL but throw an exception if the URL we request is not present or seen by the spider. `class HasHeaderContract(Contract):` `  URL = 'http://www.carzone.ie/search/Audi/A4/2.0TDI-1/49614000719014160/advert?channel=CARS'` `  def pre_process(self, response):` `  for URL in self.args:` `  if URL not in response.url:` `  raise ContractFail('URL not present')`
	Page of Pages

Test ID	LR Test 1
Purpose of Test	This test was carried out to confirm that the pandas data frame was being successfully loaded from the DB table and that the values being stored were in a useable format i.e. Strings & Ints The test will take place in three stages; 1) Verify that the DB table is successfully stored within a valid pandas data frame via psql 2) Verify that the data stored for analysis is valid and correct 3) Verify that NaN entries are replaced with 0's. 4)  This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the LR.py program file, our machine learning algorithm is not in question but the data frame from which it will gather results is being tested for its viability. https://github.com/pydata/pandas/blob/master/pandas/io/sql.py Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable. 

Test Steps	1) Open DB connection via MySQLdb 2) Select Table to pull data from 3) Create Dataframe(df) 4) Pull MYSQL table into data frame via psql.frame_query `def frame_query(*args, **kwargs):` `  """DEPRECIATED - use read_sql` `  """` `  warnings.warn("frame_query is depreciated, use read_sql", FutureWarning)` `  return read_sql(*args, **kwargs)` 5) Print data frame format to terminal to verify the data stored is correct and stored correctly
Expected Result	It is expected that the pandas data frame will connect to the DB table as specified. It is expected that the pandas data frame will populate with the correct information from the specified table. In order to vailidate, each row of the data frame is printed to terminal in a concatenated form and random rows/columns are manually verified for correctness. Anything less than 100% accuracy is unacceptable.
	Page of Pages

Test ID	LR Test 2
Purpose of Test	This test was carried out to confirm that when the LR model is being trained, it is not being corrupted by the asking price column of the pandas data frame. The test will take place in two stages; 1) Verify that the data frame being operated on is in fact broken into the data frame without the asking price, and the asking price column only for comparison. 2) Verify that the trend is accurate but not corrupted This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the LR.py program file, our machine learning algorithm accuracy is not being completely tested, rather it is being validated that our asking price column has not leaked into our LR model and corrupted our results i.e. .99% accuracy. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable. `# Create linear regression object` `LR = LinearRegression()` `# Train the model using the training sets(DataFrame without title, link or price` `LR = LR.fit(dv.transform(df_no_priceLinkTitle.T.to_dict().values()), df.price)`

Test Steps	1) Open DB connection via MySQLdb 2) Select Table to pull data from 3) Create Dataframe(df) 4) Transform dataframe into float's for LR model 5) Create two instances of data frame; Without price and only price 6) Pass both independently to LR model for training 7) Verify that price did not influence LR model accuracy by providing sanity test inputs
Expected Result	On completion, the LR model should only be as accurate as the sanity test inputs we provided. Validate that purposeful negative entities did not train accurately according to their asking price. Validate that our variance in the LR is at least 5%, if a result of over 98% is achieved, our model has been corrupted. `# Explained variance score: 1 is perfect prediction` `print ('Variance score LR: %.2f' % LR.score(dv.transform(df_no_priceLinkTitle.T.to`
	Page 1 of 1 Page

Test ID	LR Test 3
Purpose of Test	This test is to ensure that our LR prediction output is being passed successfully to our new DB table; Dealz. This step is curtail as our LR model returns a JSON encoded file and we wish to write the results in a user friendly format for the android application access later on. The test will take place in two stages; 1) Validate that the prediction output data is being written into a local JSON file for both testing and backups storage 2) Validate that our result set is being stored into the new DB table in correct format. This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the LR.py program file, our machine learning algorithm accuracy is not being tested, rather it is being validated that our resulting output data has being stored correctly in both a JSON encoded file and our relative DB table. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.  ``` data = [] with open('audi_a4.json') as f: for line in f: data.append(json.loads(line)) print data[0] ```

Test Steps	1) Run LR program 2) Verify the JSON file is created 3) Verify JSON encoded data is correct format and makes intuitive sense 4) Verify data is passed into new DB table in correct format
Expected Result	For each line in the JSON encoded file, it is expected that there will be a corresponding entry in our DB table with the information being stored correctly ie. add URL being stored in the link column as opposed to the location column.

title	link	location	C
Volkswagen Golf GTi Perfo	http://www.carzone.ie/search/Volkswagen/Golf/GTi-P...	Leinster	C

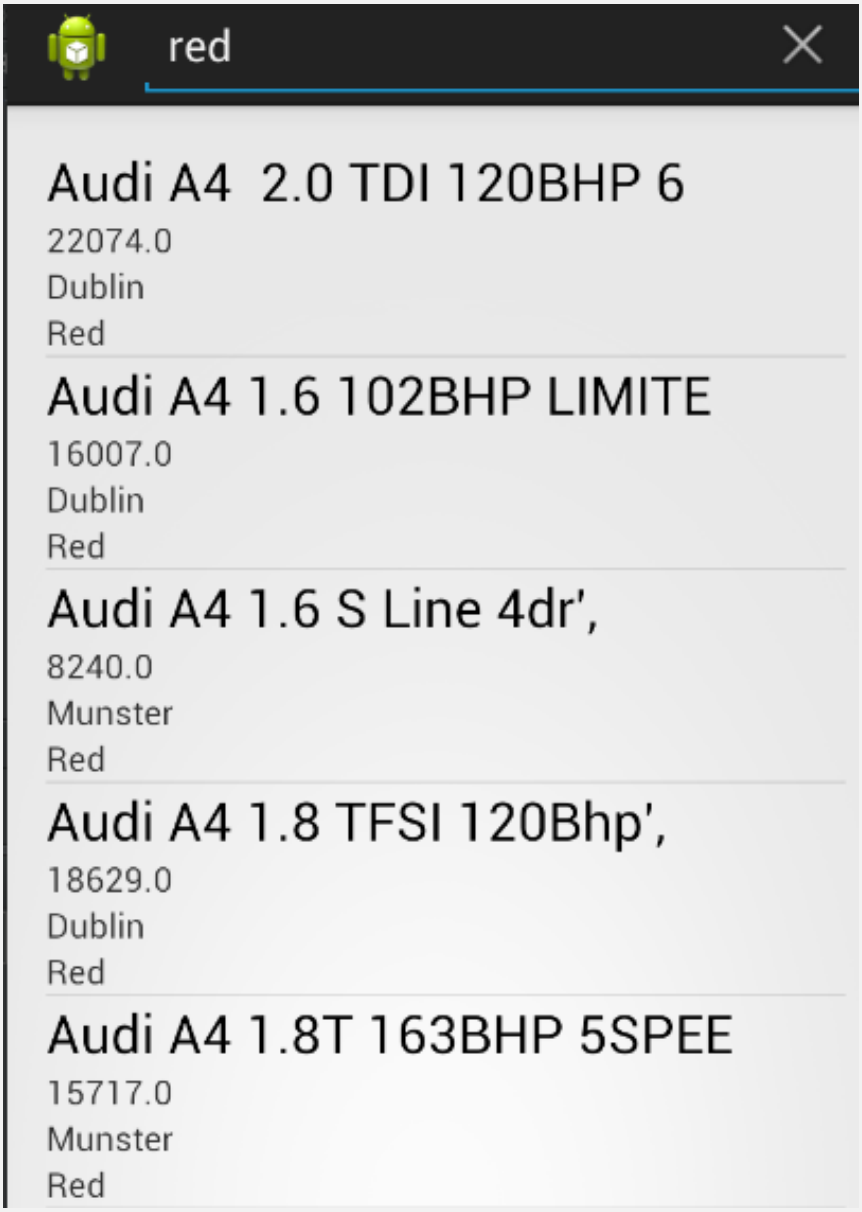
<table>
<tr><td>Test ID</td><td>LR Test 4-6</td></tr>
<tr><td>Purpose of Test</td><td>This test is to ensure that our LR prediction is roughly accurate by using sanity test cases.
This test case is the most important of all our test’s, as it will determine how close to the variance score the LR prediction is.
The test will take place in two stages;
1) Select 5 scraped adds, note their features and asking price
2) Edit add’s to convey a Great deal, Good deal, Constant, Bad deal, Awful deal
This test will contain no external dependancies to other test cases.</td></tr>
<tr><td>Test Environment</td><td>Test will take place within the context of the LR.py program file, our machine learning algorithm accuracy's being tested via sanity testing.
Test will take place on the DCU campus(L114) with a active WIFI connection.
For this test, our system state is considered stable.
<table><tr><th>title</th><th>link</th><th>price</th></tr><tr><td>BMW 3 Test Great Deal 2 35,000</td><td>http://www.carzone.ie/search/BMW/3-Series/318d-Lux...</td><td>1,000</td></tr></table></td></tr>
</table>

Test Steps	1) Select 5 scraped adds to edit 2) Set 2 to represent good deals, 2 to represent bad deals and 1 to represent a constant 3) Run LR program 4) Store output of LR 5) Verify that the prediction is accurate given the parameters provided
Expected Result	For the good and great deal test cases; Great: BMW 3 Test Great Deal 2 Predicted Price: 44071.0,Edited Asking Price: 1000.0, Original Asking Price: 42,450 Good: BMW 3 Test Good Deal 2 Predicted Price: 33,344.,Edited Asking Price: 10,000, Original Asking Price: 37,000 For the Constant test case; BMW 3 Series Constant Predicted Price: 39329.0, Original Asking Price: 35995.0 For the bad and awful deal test cases; Bad: BMW 3 Bad Deal Predicted Price: 44085.0,Edited Asking Price: 50000.0, Original Asking Price: 39,000 Awful: BMW 3 Awful Deal Predicted Price: 38277.0, Edited Asking Price: 100,000, Original Asking Price: 36,950

Test ID	Android Test 1
Purpose of Test	This test is to ensure that our the brand chosen by our user is the only brand returned within our list view for search. This step is important because our list is generated dynamically based on the model of car selected by the user. The test will take place in two stages; 1) Validate that the selected brand on click action, returns a query with the model of the car attached to filter our web server results. 2) Validate that the list returned only contains cars that belong to that brand via sanity testing. This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the brandlistfragment file, the CarService file and the bundle being passed between them. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.

Test Steps	1) Begin application 2) Select chosen brand 3) Verify that chosen brand name is sent via retro fit query to web server 4) Verify that list generated only contains cars native to that brand
Expected Result	For each of the brands selected, the web server results should filter to pull and then display the car's native to the brand selected by the user. `Intent intent = new Intent(this.getActivity(), ItemActivity.class)` `Bundle arguments2 = new Bundle();` `arguments2.putString("title", item.getTitle());` `intent.putExtra("arguements", arguments2);`
	Page of Pages

Test ID	Android Test 2
Purpose of Test	This test is to ensure that the provided search bar will return results relative to the data we have provided it with. The test will take place in two stages; 1) Validate that the search bar becomes active when selected. 2) Validate that the list returned only contains cars that belong to that criteria being searched. This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the listfragement file, the carAdapter file and the bundle being passed between them. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.
Test Steps	1) Begin application 2) Select chosen brand 3) Select search bar 4) Enter both valid and invalid search criteria

Expected Result	When the user enters valid search criteria, the application should return a valid list based on that criteria;  When the user enters invalid search criteria, the application should return only the original list.

Test ID	Android Test 3
Purpose of Test	This test is to ensure that thecae selected from the list is the same as the item we selected i.e. selected car is actually displayed correctly The test will take place in two stages; 1) Validate the user can select a car from the lost provided. 2) Validate that the item selected is displayed in a brand new fragment for the user with the correct data. This test will contain no external dependancies to other test cases.
Test Environment	Test will take place within the context of the listfragement file, the carAdapter file and the bundle being passed between them. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.
Test Steps	1) Begin application 2) Select chosen brand 3) Select a car item 4) Validate the new fragment displayed contains the car selected.
Expected Result	The item displayed should only contain data relative to the item selected;
	Page of Pages

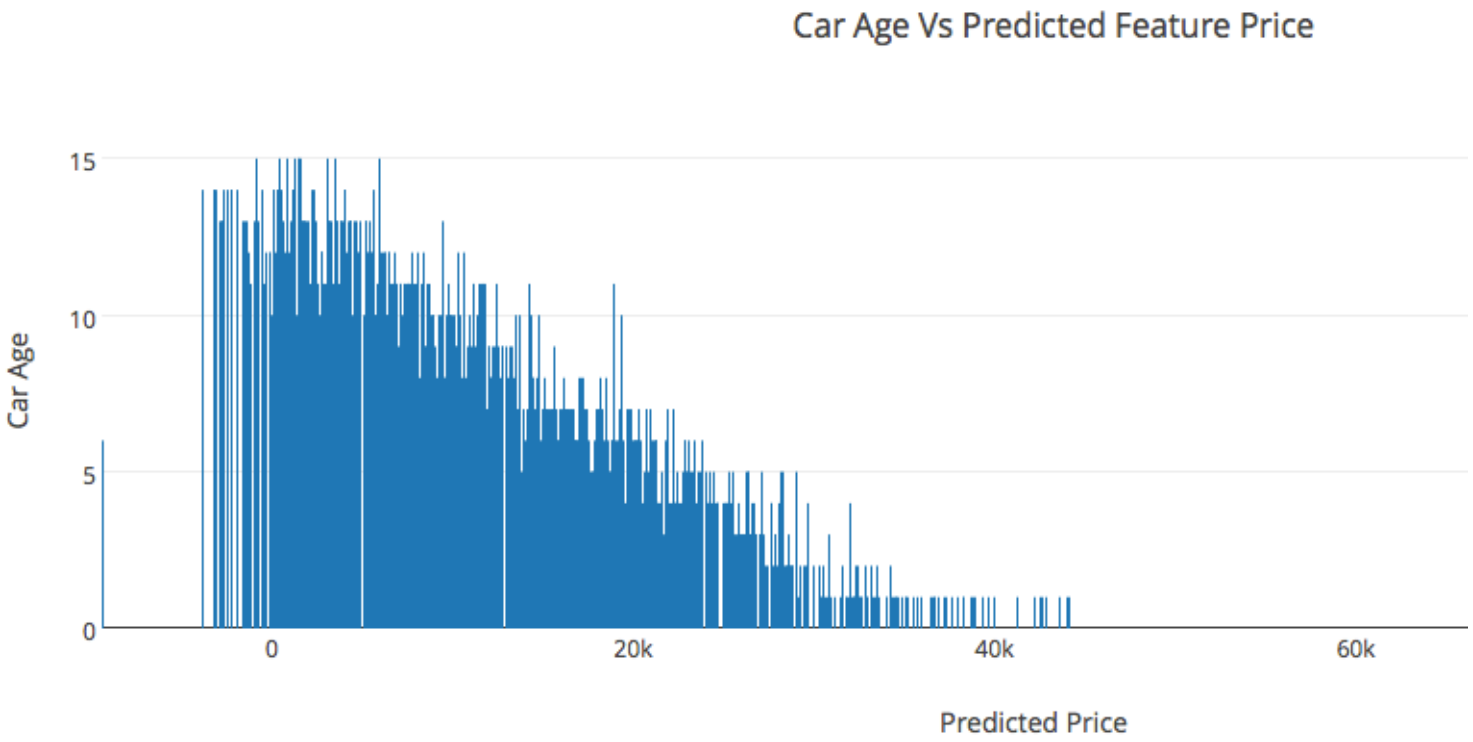
Test ID	Application Smoke Testing(3 Major Smoke Tests)
Purpose of Test	The smoke testing within my application took place in three major different stages, which we broken into 3 substages each. Mining Test: 1) Does the scraper run in the carzone.ie website? 2) Does the scraper successfully scrape the right data? 3) Does the data being scraped successfully parse into the web server? LR Test: 1) Does the LR model vectorise the data? 2) Does the LR model train? 3) Does the trained data parse back into the webserver? App Test: 1) Does the login screen pass into actual application? 2) Does the selected brand load a list of cars? 3) Does the selected item load a new window? The purpose of each section smoke test was to determine if the application build was suable for further testing and implementation. As opposed to sanity testing, my smoke tests were very general and touched large portions of the application without being too in-depth.
Test Environment	Test will take place within the context of all parts of the application. The purpose of the test's is to touch each aspect of the application broadly to evaluate if we can continue with the sanity/unit tests. Test will take place on the DCU campus(L114) with a active WIFI connection. For this test, our system state is considered stable.

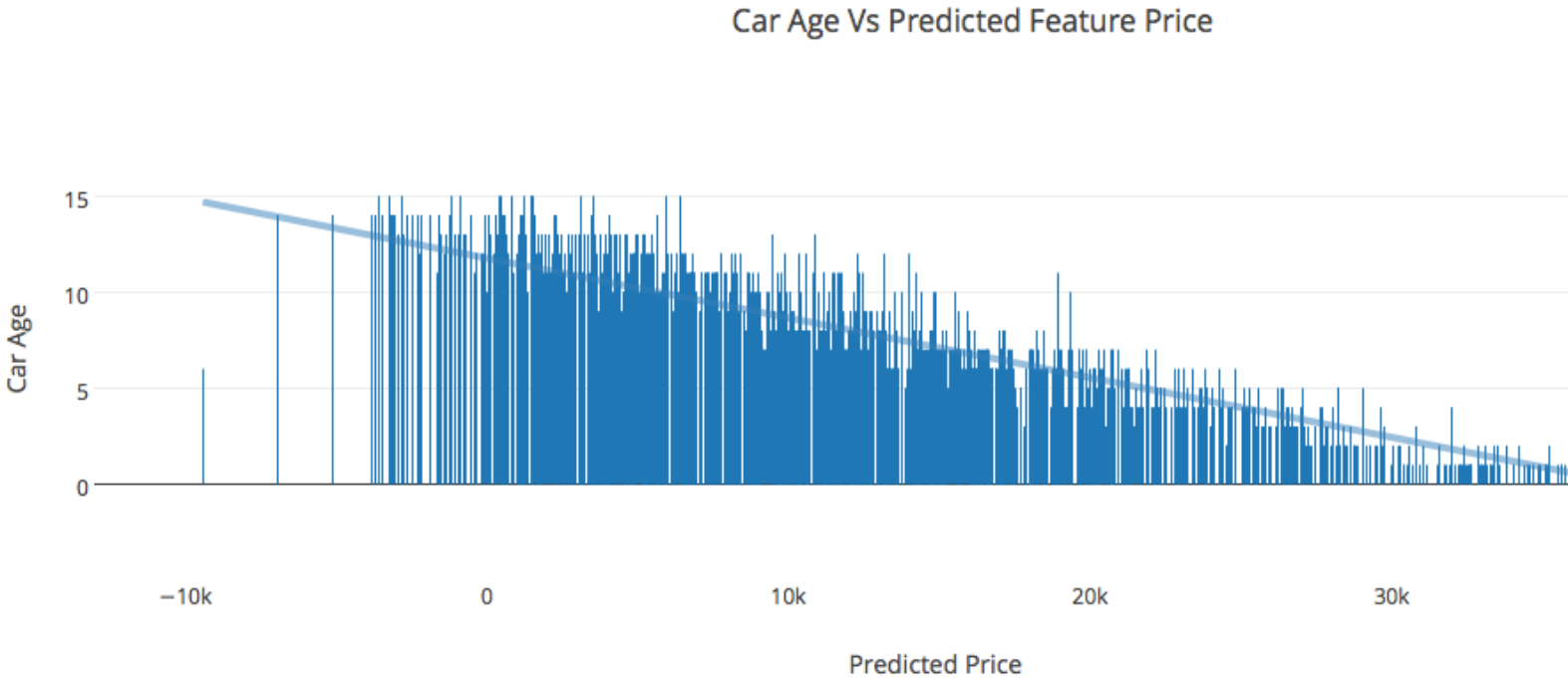
Test Steps	1) Mining Test 1: Run the scraper with the start URL aimed at carzone.ie Test 2: Validate data scraped manually in small cross sections Test 3: Validate that the data is stored successfully in the web server 2) Machine Learning Test 1: Test the DictVectoriser for accuracy Test 2: Run LR model fit and see if it does train Test 3: Validate that the data is stored successfully in the web server 3) Android Test 1: Validate log in screen Test 2: Validate if brand selected returns valid list Test 3: Validate when a item is selected a new item window is loaded
Expected Result	It is expected that within each test performed, the system build will perform in a stable enough manor for the project development/Testing to continue.

Graph Representation Per Feature: LR.py

All features graphed below are a visual representation of how our features are scored within the LR model. What we expect to see in these visual test cases is a demonstration of accuracy i.e. As the mileage figure Varies, the associated prediction should represent a uniform average.

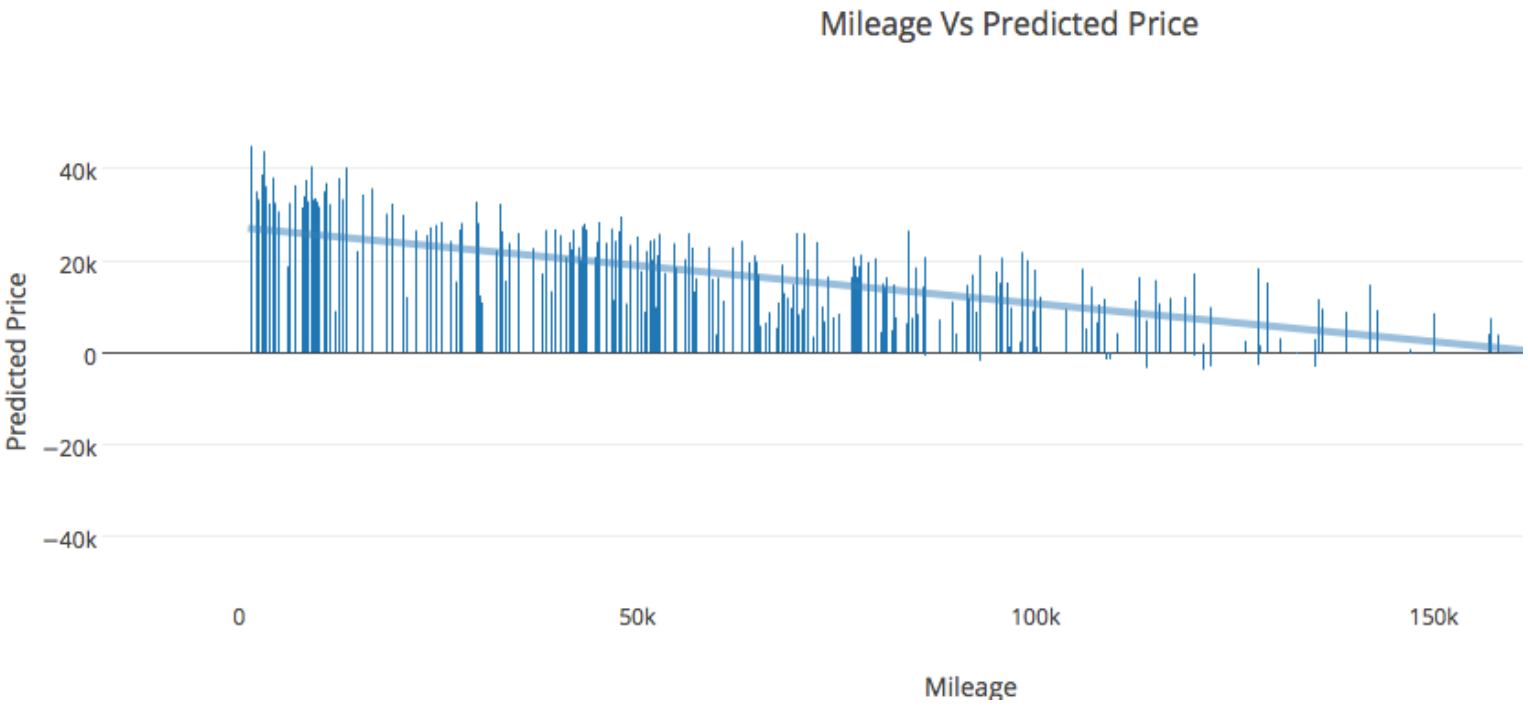
Car Age:





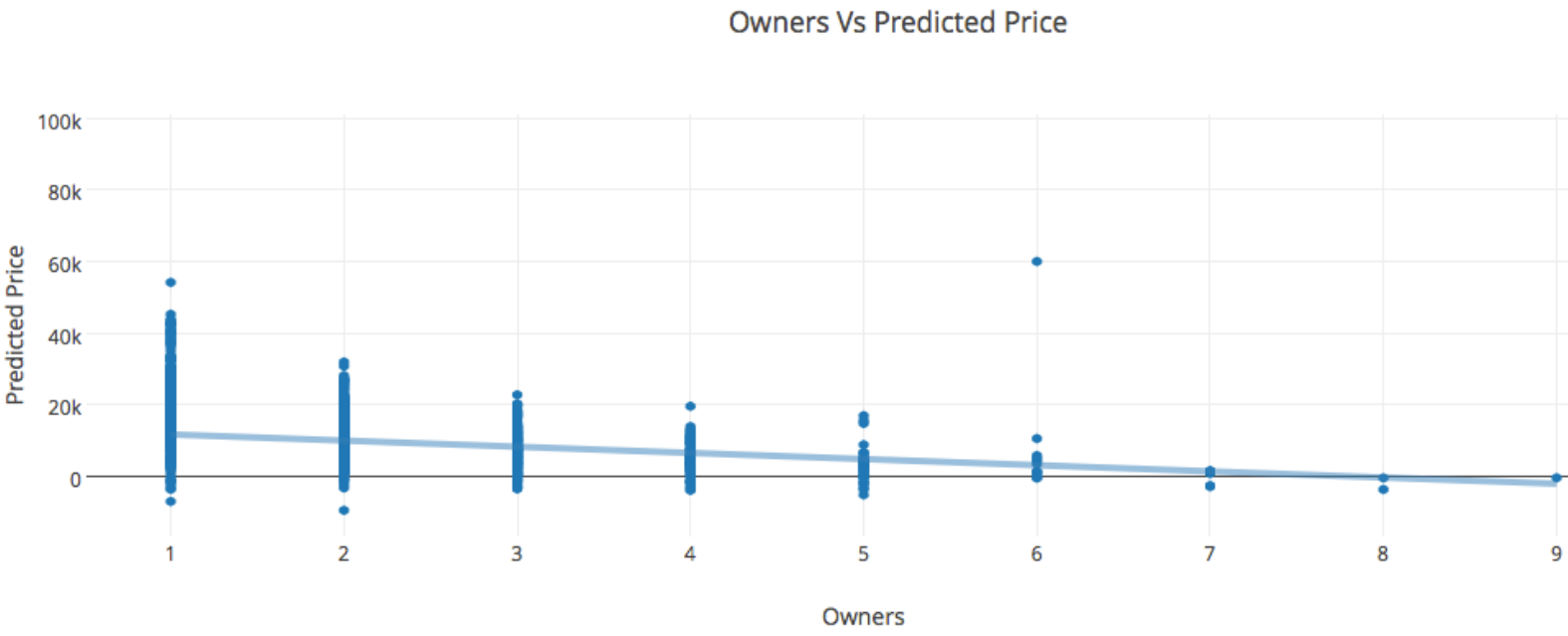
From the graphs, we can see that as our car age increases, the associated prediction for that feature decreases in value as the car age decreases i.e. the closer to 2014, the more valuable that feature is deemed.

Mileage:



From the graph, we can see that as our mileage increases, the associated prediction for that feature decreases in value & vice versa i.e. the LOWER the mileage of a car, the more valuable that feature is deemed.

Owners:



From the graph, we can see that as our car owners increase, the associated prediction for that feature decreases in value & the prediction that take place via the Linear regression fit.