Zillow

Web scraping & House Price Prediction using TensorFlow

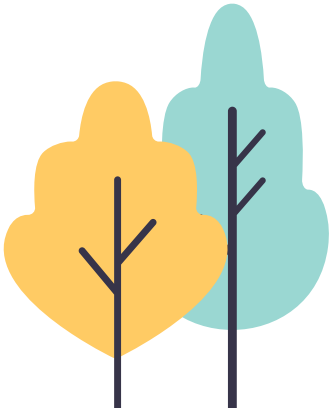
Group Assignment 4

Course: ALY 6040 Data Mining
Instructor: Fidel Rodriguez

Group - 1
Pragati Koladiya | NUID: 00102944
Priyanka Kanukollu | NUID: 001021111
Tanvi Bhagat | NUID: 001083830
Ketaki Joshi | NUID: 001069747

Goals

- Introduction
- Web scraping from Zillow website and store data in database
- Fetch the data for house price description
- Models implementation using TensorFlow
- Comparisons of models
- Conclusion



Dataset Information

- Database connection established and created tables to store the data.

```
def establish_database_connection(attempts = 0):
  if attempts >4:
    print("Error in establishing connection...  tried 5 times now quitting")
    return -1
  else:
    try:
     cnx = mysql.connector.connect(user='root4', password='!GoodPassword123!',
                                   host='34.67.144.96',
                                   port=3306,
                                   database='test_schema',
                                   auth_plugin='mysql_native_password')
    except:
      print("Error in establishing connection...  snoozing for 2 mins and retrying")
      time.sleep(120)
      establish_database_connection(attempts+1)
  return cnx
```

- Created 4 tables in MySQL workbench for storing the house data.

```
#Checking created tables in the database
cnx = establish_database_connection()

query = 'SHOW TABLES;'

cursor = cnx.cursor()
cursor.execute(query)
record_counter = 0
print("Begin output")
for i in cursor:
  print("Record #", record_counter, ": ", i)
  record_counter = record_counter + 1
print("End output")
cnx.commit()
cursor.close()
cnx.close()
```

```
Begin output
Record # 0 :  ('house_details_results_preliminary',)
Record # 1 :  ('house_details_results_work',)
Record # 2 :  ('house_details_workers',)
Record # 3 :  ('new_page_house_description',)
End output
```

ZILLOW HOME

- The data is being collected for six different zip codes which are used in the following query

```
query = """
INSERT INTO house_details_results_work (zip)
VALUES
(91325), (91326), (91329), (91331), (91335), (90042);
"""
```

- Data is being scraped from Zillow website using BeautifulSoup.

```
for z in zip_code_list:

  url_list = []
  request_list = []
  soup_list = []
  error_flag = 0
  final_soup_index = 20
  for i in range(20):
    if i > final_soup_index:
      continue
    if i != 0:
      random_wait()
    url_list.append('https://www.zillow.com/homes/'+str(z)+'/'+str(i+1)+'_p/')
    request_list.append(requests.Session().get(url_list[i], headers=req_headers))
    soup_list.append(BeautifulSoup(request_list[i].content, 'html.parser'))
```

- Removed HTML Tags using split function.

```
for i in range(0,len(df)):
  if df.beds[i].find('li class="">') != -1 and df.beds[i].find('<abbr class="list-card-label"> <!-- -->bds') != -1:
    df.beds[i] =  df.beds[i].split('<li class="">')[1].split('<abbr class="list-card-label"> <!-- -->bds')[0]

  elif df.beds[i].find('<li class="">') != -1 and df.beds[i].find('<abbr class="list-card-label"> <!-- -->bd') != -1:
    df.beds[i] =  df.beds[i].split('<li class="">')[1].split('<abbr class="list-card-label"> <!-- -->bd')[0]
  else:
    df.beds[i] = 0

  if df.baths[i].find('>bds</abbr>,</li><li class="">') != -1 and df.baths[i].find('<abbr class="list-card-label"> <!-- -->ba') != -1:
    df.baths[i] =  df.baths[i].split('>bds</abbr>,</li><li class="">')[1].split('<abbr class="list-card-label"> <!-- -->ba')[0]
  elif df.baths[i].find('>bd</abbr>,</li><li class="">') != -1 and df.baths[i].find('<abbr class="list-card-label"> <!-- -->ba') != -1:
    df.baths[i] =  df.baths[i].split('>bd</abbr>,</li><li class="">')[1].split('<abbr class="list-card-label"> <!-- -->ba')[0]
  else:
    df.baths[i] = 0

  if df.sqft[i].find('ba</abbr>,</li><li class="">') != -1 and df.sqft[i].find('<abbr class="list-card-label"> <!-- -->sqft') != -1:
    df.sqft[i] =  df.sqft[i].split('ba</abbr>,</li><li class="">')[1].split('<abbr class="list-card-label"> <!-- -->sqft')[0]

  elif df.sqft[i].find('ba</abbr>,</li><li class="">') != -1 and df.sqft[i].find('<abbr class="list-card-label"> <!-- -->sqft lot') != -1:
    df.sqft[i] =  df.sqft[i].split('ba</abbr>,</li><li class="">')[1].split('<abbr class="list-card-label"> <!-- -->sqft lot')[0]
  else:
    df.sqft[i] = 0

  if df.status[i].find('<li class="list-card-statusText">-') != -1 and df.status[i].find('</li></ul>') != -1:
    df.status[i] =  df.status[i].split('<li class="list-card-statusText">-')[1].split('</li></ul>')[0]
  else:
    df.status[i] = 'N/A'
```

- Created the main function to call the helper functions:

```
def main():
  initialize_database_environment()
  setup_database_tables()
  get_and_set_worker_id()

  #Setup environment variables:
  workload_limit = 6
  processed_records = 0

  #Initialize the worker / start telemetry
  my_worker_id, my_ip_address, my_start_time = get_and_set_worker_id()
  print("My worker_id: ",my_worker_id,"; My IP Address: ", my_ip_address, "; My Start Time: ", my_start_time)

  #Get a worklist
  work_list = lock_and_get_work_list(my_worker_id, workload_limit)
  print("Work list is:", work_list)

  #Process the worklist
  analysis_results, analysis_record_count = execute_query_and_return_results(work_list, my_worker_id)
  print("Processed",analysis_record_count,"records.")

  #Write the worklist to the DB
  processed_records += insert_results_to_db(analysis_results)
```

- Stored the data (list price, address, zip codes, beds, baths, status and links) into the MySQL Workbench by fetching it from Zillow website.

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	worker_id	list_price	address	zip	beds	baths	sqft	status	link
1	2	435000	5059 1/2 Shiplev Glen Dr Los Angeles CA 90042	90042	2	1	720	Condo for sale	https://www.zillow.com/homedetails/5059-1-2-Shiplev-Glen-Dr-Los-Angeles-CA-90042/207338228...
2	2	895000	118 S Avenue 50 UNIT 601 Los Angeles CA 90042	90042	2	3	1370	Townhouse for sale	https://www.zillow.com/homedetails/118-S-Avenue-50-UNIT-601-Los-Angeles-CA-90042/3027962...
3	2	1395000	5055 Meridian St Los Angeles CA 90042	90042	4	3	1968	House for sale	https://www.zillow.com/homedetails/5055-Meridian-St-Los-Angeles-CA-90042/20766341 zpid/
4	2	599000	258 S Avenue 52 Los Angeles CA 90042	90042	2	1	924	House for sale	https://www.zillow.com/homedetails/258-S-Avenue-52-Los-Angeles-CA-90042/20761929 zpid/
5	2	789000	979 Dexter St Los Angeles CA 90042	90042	3	1	951	House for sale	https://www.zillow.com/homedetails/979-Dexter-St-Los-Angeles-CA-90042/20767040 zpid/
6	2	750000	4225 Via Arbolada UNIT 511 Los Angeles CA 90...	90042	2	3	1682	Townhouse for sale	https://www.zillow.com/homedetails/4225-Via-Arbolada-UNIT-511-Los-Angeles-CA-90042/206882...
7	2	630000	4049 Via Marisol APT 116 Los Angeles CA 90042	90042	2	2	1183	Condo for sale	https://www.zillow.com/homedetails/4049-Via-Marisol-APT-116-Los-Angeles-CA-90042/20688005 ...
8	2	650000	Via Marisol Los Angeles CA 90042	90042	2	3	1460	Coming soon	https://www.zillow.com/homedetails/5511-Via-Marisol-101-Los-Angeles-CA-90042/2073551867 zpid/
9	2	715000	5777 Aldama St Los Angeles CA 90042	90042	2	2	1073	House for sale	https://www.zillow.com/homedetails/5777-Aldama-St-Los-Angeles-CA-90042/20770382 zpid/
10	2	969093	5327 Meridian St Los Angeles CA 90042	90042	3	2	1681	House for sale	https://www.zillow.com/homedetails/5327-Meridian-St-Los-Angeles-CA-90042/20768142 zpid/
11	2	3995000	1443 N Avenue 47 Los Angeles CA 90042	90042	0	0	0	Multi-family home f...	https://www.zillow.com/homedetails/1443-N-Avenue-47-Los-Angeles-CA-90042/2098081781 zpid/
12	2	536000	519 Neva Pl Los Angeles CA 90042	90042	2	1	669	House for sale	https://www.zillow.com/homedetails/519-Neva-Pl-Los-Angeles-CA-90042/20771637 zpid/
13	2	1088000	118 S Avenue 50 UNIT 203 Highland Park CA 90...	90042	3	3	2140	Condo for sale	https://www.zillow.com/homedetails/118-S-Avenue-50-UNIT-203-Highland-Park-CA-90042/209240...
14	2	939000	5925 Terrace Dr Los Angeles CA 90042	90042	2	1	1036	House for sale	https://www.zillow.com/homedetails/5925-Terrace-Dr-Los-Angeles-CA-90042/20770530 zpid/
15	2	699900	5050 Irvington Pl Los Angeles CA 90042	90042	0	0	0	Multi-family home f...	https://www.zillow.com/homedetails/5050-Irvington-Pl-Los-Angeles-CA-90042/20762323 zpid/

Implemented Try & Catch block while fetching data

- Handling exception while fetching the list of links for different zip codes from the database.

```
def execute_query_and_return_results():
  record_count = 0
  cnx = establish_database_connection()
  new_page = []
  query = ("""SELECT link FROM house_details_results_preliminary;""")
  try:
   cursor = cnx.cursor()
   cursor.execute(query)
   for i in cursor:
    record_count += 1
    new_page.append(i)
   cnx.commit()
   print(record_count, "records fetched.")
  except:
   print("Error occured when fetching records")
  finally:
    cursor.close()
    cnx.close()

  return new_page
```

```
new_page = execute_query_and_return_results()
```

```
442 records fetched.
```

- Obtained the results from the database and stored it in the new data frame.

```
df = pd.DataFrame(new_page, columns = ['new_page'])
```

```
df.head()
```

	new_page
0	https://www.zillow.com/homedetails/5059-1-2-Shipley-Glen-Dr-Los-Angeles-CA-90042/2073382282_zpid/
1	https://www.zillow.com/homedetails/118-S-Avenue-50-UNIT-601-Los-Angeles-CA-90042/302796219_zpid/
2	https://www.zillow.com/homedetails/5055-Meridian-St-Los-Angeles-CA-90042/20766341_zpid/
3	https://www.zillow.com/homedetails/258-S-Avenue-52-Los-Angeles-CA-90042/20761929_zpid/
4	https://www.zillow.com/homedetails/979-Dexter-St-Los-Angeles-CA-90042/20767040_zpid/

- Implemented the for loop to fetch the link of each house cards and grabbed the data for house price description.

```
#storing new_page list into new_page_link
new_page_link = df['new_page']
```

```
#new_page_link
url_list1 = [] #relocated to for loop
request_list1 = [] #relocated to for loop
soup_list1 = [] #relocated to for loop

#processed_records for number of records to pass dynamic record count
for i in range(0,len(df)):
  url1 = new_page_link[i]
  request = requests.Session().get(url1, headers=req_headers)
  soup = BeautifulSoup(request.content, 'html.parser')
  url_list1.append(url1)
  request_list1.append(request)
  soup_list1.append(soup)
```

- Fetched the required data and removed the HTML tag.

```
de = []
de_pay = []
de_img = []

for soup in soup_list1:
  description = soup.find_all("div", {"class" : "ds-overview-section"})
  de.append(description)
  Est_pay = soup.find_all("div", {"class" : "sc-qPlga gQjcXU ds-chip-removable-content"})
  de_pay.append(Est_pay)
  img = soup.find_all("div", {"class" : "hdp-photo-carousel"})
  de_img.append(img)
```

- Implemented the for loop to fetch the link of each house cards and grab the data for house price description.

```
for i in range(0,len(df_new1)):
  if df_new1.description[i].find("sc-ptDSg") != -1 and df_new1.description[i].find("</div><button class=") != -1:
    df_new1.description[i] =  df_new1.description[i].split('sc-ptDSg')[1].split('</div><button class=')[0]
    df_new1.description[i] =  df_new1.description[i][10:]
  else:
    if len(df_new1.description[i]) < 3 :
     print("description is empty  for row: " + str(i))
    else:
     print("description is special for row: " + str(i))
```

- Fetched the Zestimate price and removed HTML tags.

```
for i in range(0,len(df_new1)):
  # By splitting the data to fetch the required content
  if df_new1.Est_pay[i].find('<span class="Text-c11n-8-18-0__aiai24-0 einFCw">') != -1 and df_new1.Est_pay[i].find("</span>") != -1:
    df_new1.Est_pay[i] =  df_new1.Est_pay[i].split('<span class="Text-c11n-8-18-0__aiai24-0 einFCw">')[1].split('</span>')[0]
    df_new1.Est_pay[i] =  df_new1.Est_pay[i][0:]
  else:
    # Else it will show the empty row with allocated number, if there are lessthan 3 string values.
    df_new1.Est_pay[i] = 'N/A'
```

- Removed HTML tags from image

```
for i in range(0,len(df_new1)):
  # By splitting the data to fetch the required content
  if df_new1.img[i].find('"photo-tile-image" ') != -1 and df_new1.img[i].find('"/></button>') != -1:
    df_new1.img[i] =  df_new1.img[i].split('"photo-tile-image" ')[1].split('"/></button>')[0]
    df_new1.img[i] =  df_new1.img[i][5:]
  else:
    # Else it will show the empty row with allocated number, if there are lessthan 3 string values.
    df_new1.img[i] = 'N/A'
```

Implemented Try & Catch block while inserting data

- Stored the house data in MySQL Workbench.
- Handled exception while inserting the house description.

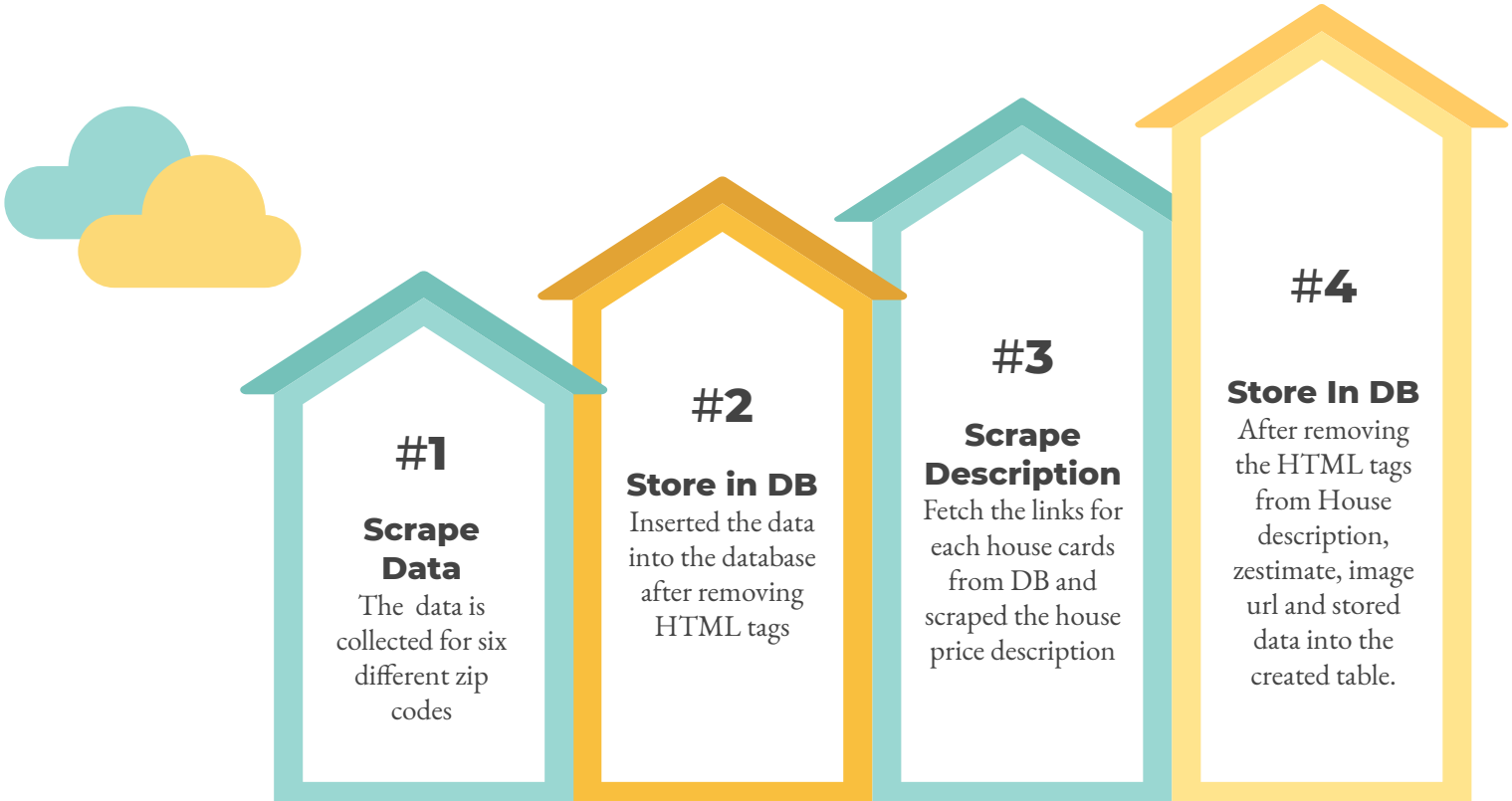
```
def insert_results_to_db(df_new1):
  cnx = establish_database_connection()
  cursor = cnx.cursor()
  record_count = 0
  cols = ",".join([str(i) for i in df_new1.columns.tolist()])
  try:

    for i, row in df_new1.iterrows():
      sql = "INSERT INTO new_page_house_description (" + cols + ") VALUES (" + "%s,"*(len(row)-1) + "%s)"
      cursor.execute(sql, tuple(row))
      cnx.commit()
      record_count += 1
  except:
    print("Error occured while inserting house description in DB")
  finally:
   cnx.close()
  return record_count
```

- Displayed the records of house descriptions stored in the database

```
cnx = establish_database_connection()

query = 'SELECT * FROM new_page_house_description LIMIT 5;'

cursor = cnx.cursor()
cursor.execute(query)
record_counter = 0
print("Begin output")
for i in cursor:
  print("Record #", record_counter, ": ", i)
  record_counter = record_counter + 1
print("End output")
cnx.commit()
cursor.close()
cnx.close()
```

```
Begin output
Record # 0 :  (1, " Nestled on the hillside in a great Highland Par
Record # 1 :  (2, " NELA's (NorthEast LA) best location - Highland
Record # 2 :  (3, ' Newly Remodeled and Expanded Highland Park Oasi
Record # 3 :  (4, ' For the Urbanite who likes living within close
Record # 4 :  (5, 'Gorgeous, private, creative retreat.1925 charact
End output
```

Workflow review before building TensorFlow



#1

Scrape Data

The data is collected for six different zip codes

#2

Store in DB

Inserted the data into the database after removing HTML tags

#3

Scrape Description

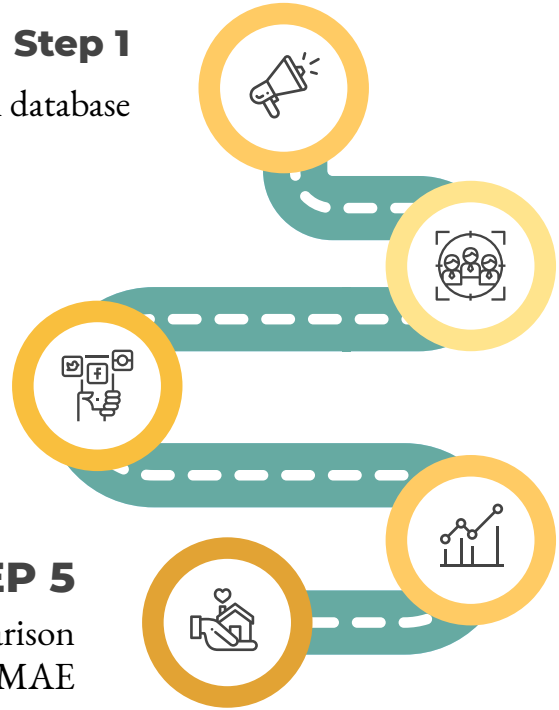
Fetch the links for each house cards from DB and scraped the house price description

#4

Store In DB

After removing the HTML tags from House description, zestimate, image url and stored data into the created table.

Steps Followed For TensorFlow Model Implementation



Step 1

Fetched data from database

Step 2

Created dummy variables for zip-codes

Step 3

Normalized the data

Step 4

Implemented 4 models using TensorFlow.

STEP 5

Model Comparison based on MAE

Normalizing the data

```
#implemented the normalization for single variable SQFT
data_normalizer_sqft = preprocessing.Normalization()
data_normalizer_sqft.adapt(np.array(df_train['sqft']))
print(data_normalizer_sqft.mean.numpy())
```

```
[1931.057]
```

```
#implemented the normalization for multiple variables
data_normalizer = preprocessing.Normalization()
data_normalizer.adapt(np.array(df_train))
print(data_normalizer.mean.numpy())
```

```
[   3.374    2.875 1931.057    0.185    0.07     0.195    0.3      0.118
    0.131]
```

- By using the normalization it makes the training data much more stable.
- By giving the scaled inputs, will get the scaled outputs.

Implemented TensorFlow

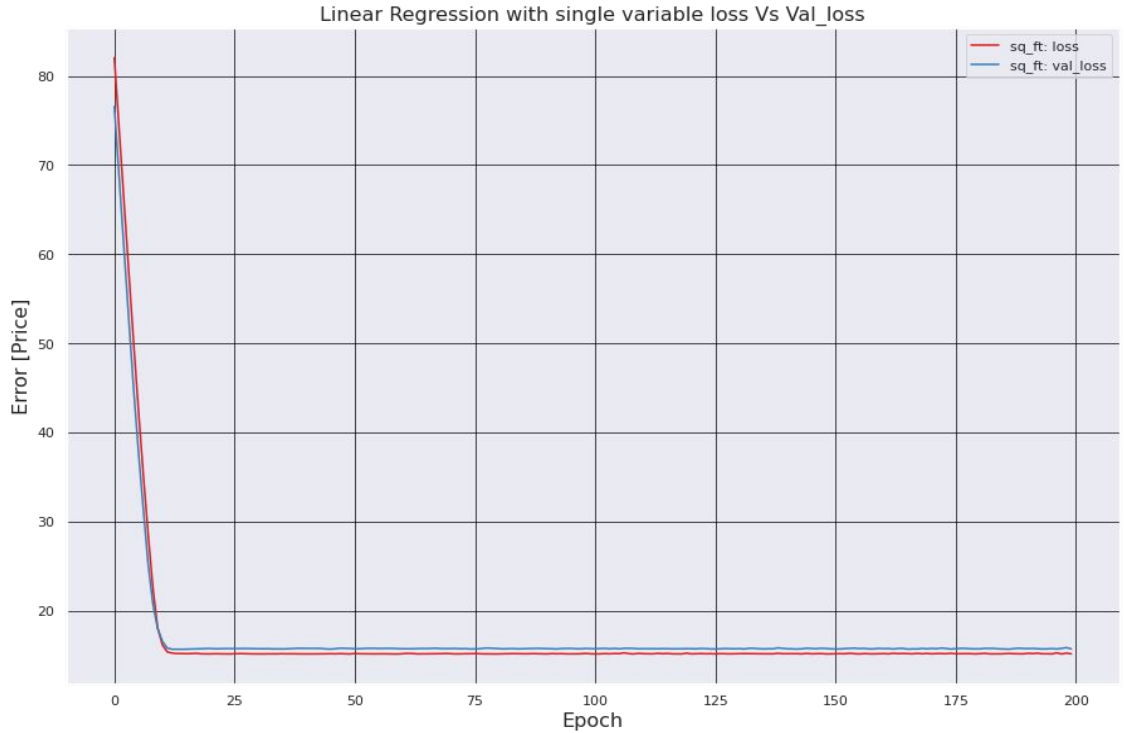
Linear Regression with multiple variables

Deep Neural Network (DNN) with single variable

Linear Regression with single variable

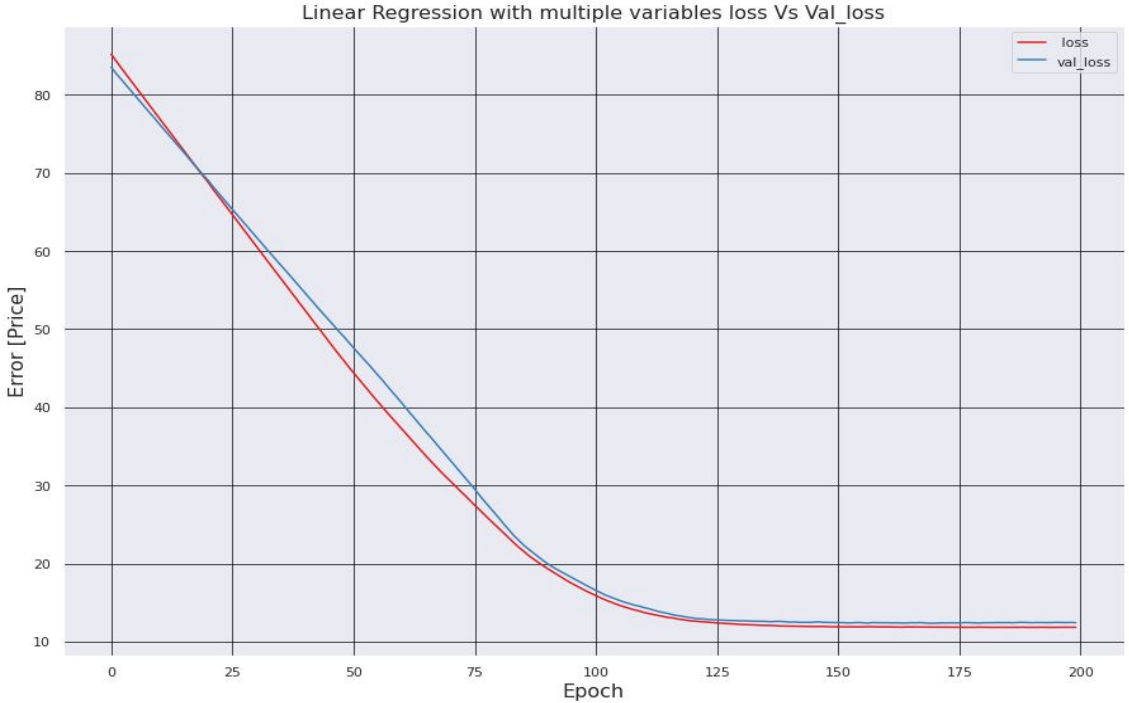
Deep Neural Network (DNN) with multiple variables

Linear regression using single variable (Square feet)



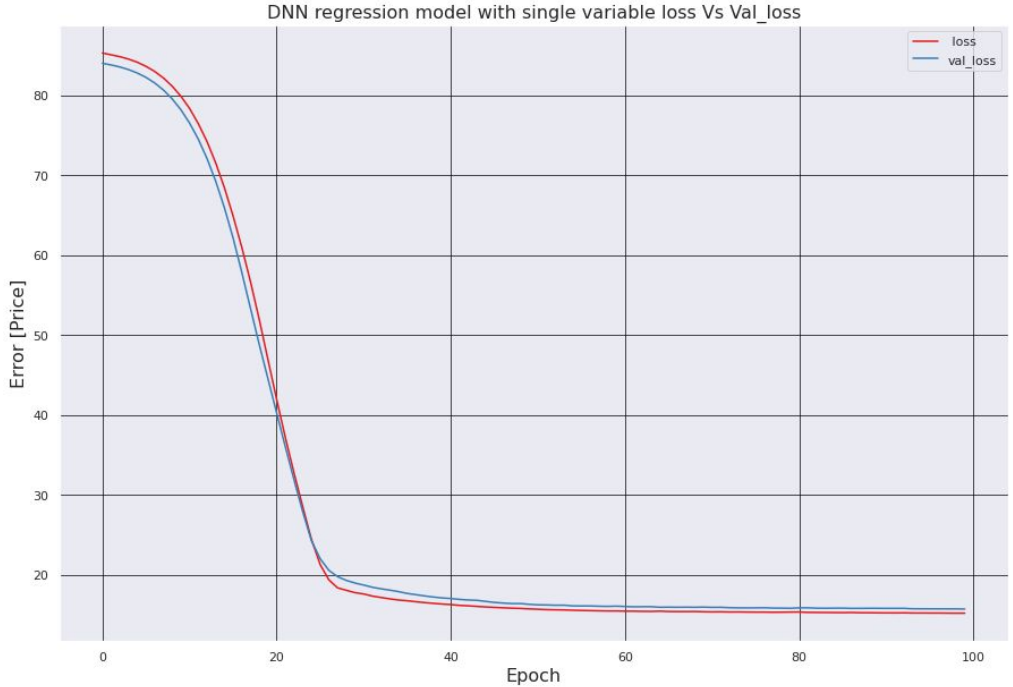
- After the model is built, configured the training procedure using model.compile().
- Once the training is configured, fit the model to execute the training.

Linear regression for multiple variables (beds, bath, square feet)



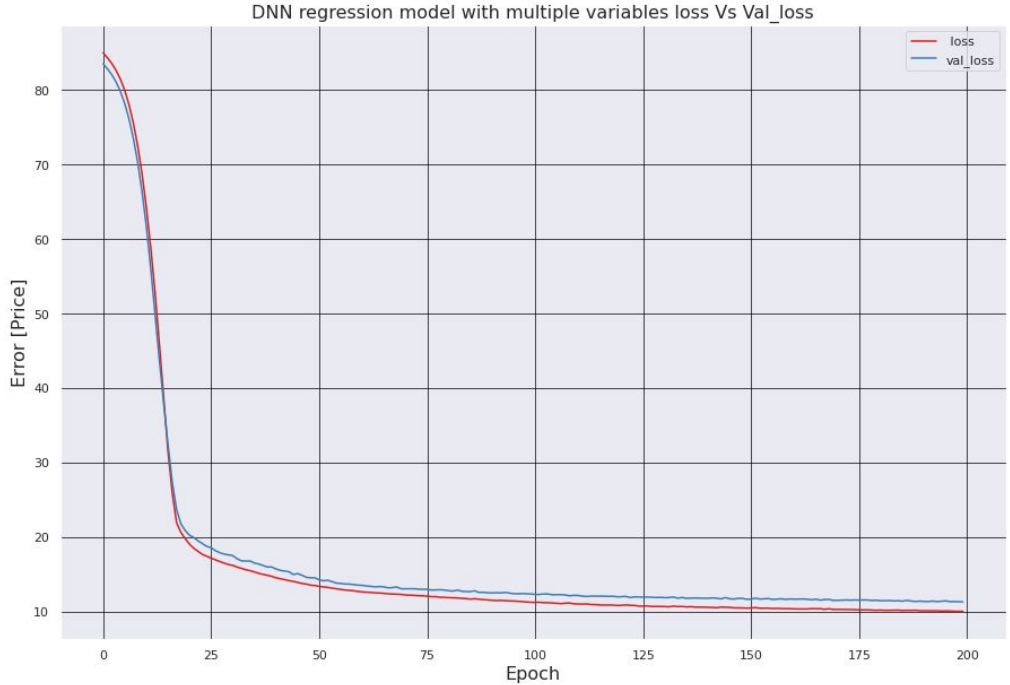
- Predicting the price using multiple variables.
- Using adam optimizer with epochs =200 and validation split = 0.2

DNN for single variable (Square feet)



- In the DNN regression after normalization layer, there are two hidden, nonlinear, dense layers (using the relu nonlinearity) are implemented.
- For compile we used build_and_compile_model function to configure training procedure.

DNN regression for multiple variables (beds, bath, square feet)



- In the DNN regression after normalization layer, there are Two hidden, nonlinear, Dense layers (using the relu nonlinearity) are implemented.
- For compiling, we used build_and_compile_model function to configure training procedure.

Model Comparison

Based on Mean Absolute error (MAE), DNN Regression – Multiple variables is the best model that can be used for house price prediction.

```
pd.DataFrame(test_results, index=['Mean absolute error']).T
```

	Mean absolute error
house_price_linear_model_sqft	13.364033
house_price_linear_model2	11.536201
dnn_house_price_model	13.069312
dnn_house_price_model2	10.614470

House Price Prediction

```
[ ] df_predict.columns

    Index(['zip', 'beds', 'baths', 'sqft'], dtype='object')
```

```
[ ] #Creating the dummy variables for six zipcodes
    dummy=pd.get_dummies(df_predict['zip'])

    #Concated the dummy variables to main file
    df_predict=pd.concat([df_predict,dummy],axis=1)

    #After creating the dummy variables, we dropped the Zip column
    df_predict = df_predict.drop(columns = ['zip'], axis = 1)
```

```
[ ] df_predict.head()
```

	beds	baths	sqft	90042	91325	91326	91329	91331	91335
0	2	1.0	720	1	0	0	0	0	0
1	2	3.0	1370	1	0	0	0	0	0
2	4	3.0	1968	1	0	0	0	0	0
3	2	1.0	924	1	0	0	0	0	0
4	3	1.0	951	1	0	0	0	0	0

- Used the actual data to predict the house price
- Created dummy variables for zip codes.

- Arranged data for price prediction

```
X_dummy = df_predict
y_dummy_pred = dnn_house_price_model2.predict(X_dummy)
```

- Predicted the house price using best performing model – DNN regression (multiple variables)

```
prediction_dummy_df = pd.DataFrame(dnn_house_price_model2.predict(X_dummy), columns = {'Predicted_List_Price (x10K)'}).set_index([pd.Index(X_dummy.index)])
all_dummy_df = X_dummy.join(prediction_dummy_df)
all_dummy_df
```

	beds	baths	sqft	90042	91325	91326	91329	91331	91335	Predicted_List_Price (x10K)
0	2	1.0	720	1	0	0	0	0	0	68.939461
1	2	3.0	1370	1	0	0	0	0	0	76.144524
2	4	3.0	1968	1	0	0	0	0	0	115.514427
3	2	1.0	924	1	0	0	0	0	0	76.243233
4	3	1.0	951	1	0	0	0	0	0	87.073708
5	2	3.0	1682	1	0	0	0	0	0	88.576271

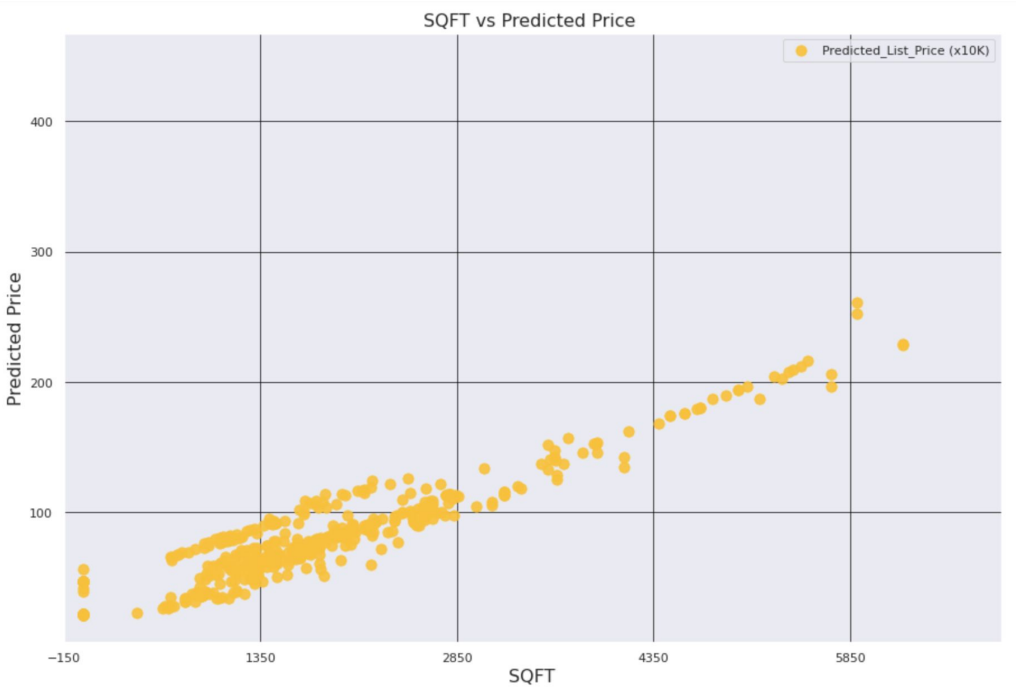
Combined data for Application

- Created 3 csv files(house details, house description and predicted house price) and merged them together

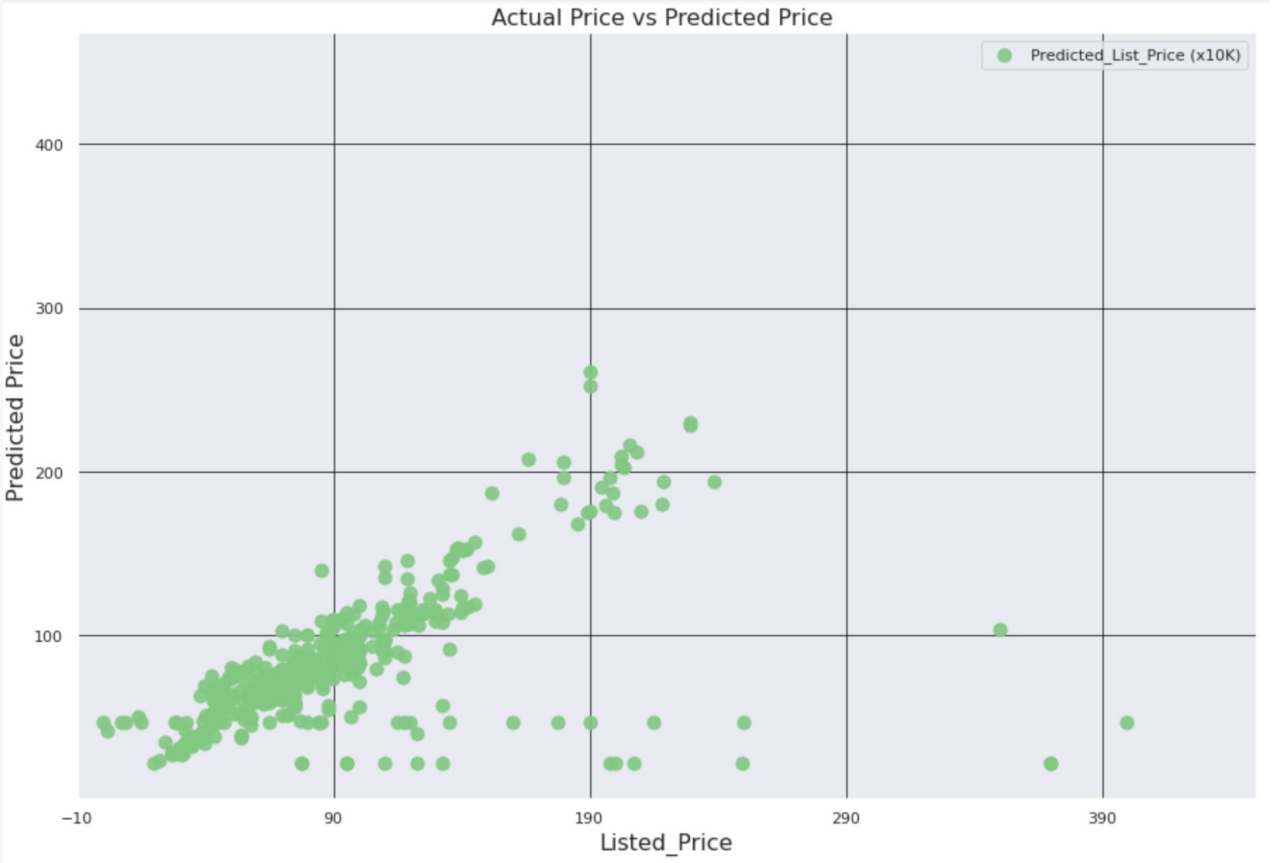
```
# reading all csv files
data1 = pd.read_csv('/content/Home.csv')
data2 = pd.read_csv('/content/SecondHome.csv')
data3 = pd.read_csv('/content/Prediction.csv')

# using merge function by setting how='inner'
combined_df = pd.merge(data1, data2)
ZillowData = pd.merge(combined_df,data3)
```

```
ZillowData.to_csv('Zillow.csv')
```

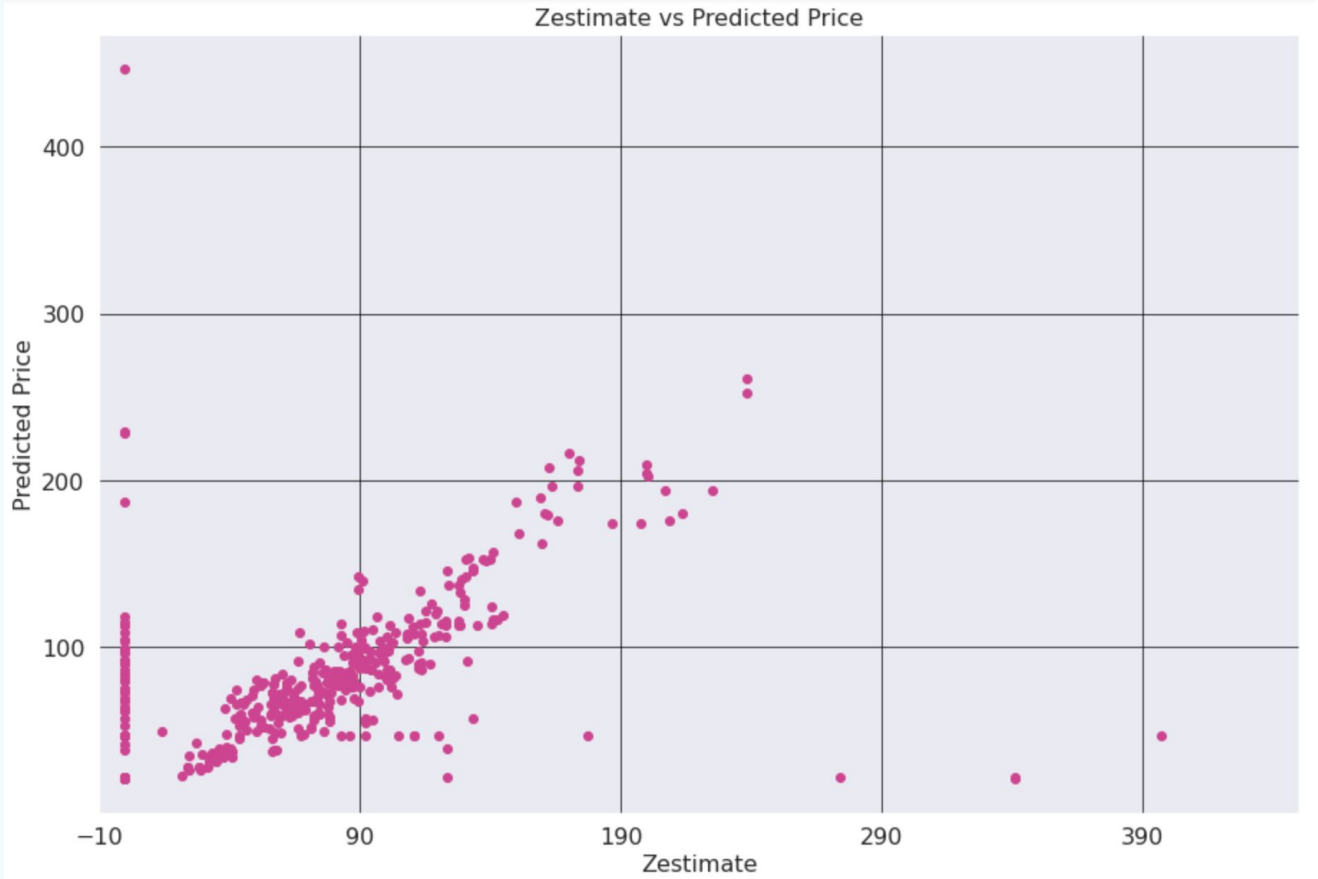
Analysis for predicted house price



- The above graph represents square feet versus predicted price(x10k) with positive correlation. As square feet increases, the predicted price is also increasing.



- The above graph represents actual listed price versus predicted price in (x10k).



- The above graph represents Z-estimate versus predicted price.

- Predicted for six different zip codes using the user input data

```
def price_predict_ip_param(ip_beds,ip_baths,ip_sqft ):
  ip_zip =  [  90042,  91325,  91326,  91329,  91331,  91335 ]
  data = [[ip_beds, ip_baths  , ip_sqft, 0, 0 , 0 , 0 , 0, 0 ]]
  ip_df = pd.DataFrame(data, columns = ['beds' , 'baths', 'sqft',  90042,  91325,  91326 , 91329,  91331,  91335])

  for i in range(len(ip_zip)+1):
    #print(ip_df)
    if i >0:
      #print('\n')
      #print(i)
      #print(ip_df)
      ip_df.loc[len(ip_df.index)] = ip_df.iloc[0]
      ip_df.loc[i][ip_zip[i-1]] =  1
  ip_df = ip_df.drop([0])

  ip_df['predicted_price(x10k)'] =  dnn_house_price_model2.predict(ip_df)
  return (ip_df)
```

Recommended best price according to user requirements across six zip codes

```
#User prompt to enter the requirements based on number of bedrooms, bathrooms and squarefeet
ip_beds  = input ("Enter a number of Beds: ")
ip_baths = input ("Enter a number of Baths: ")
ip_sqft  = input ("Enter a number of sqft: ")

result =  price_predict_ip_param(int(ip_beds),int(ip_baths),int(ip_sqft))

# Prints in the console the variable as requested
print (result)

#Showing the minimum price prediction value
res_min = min(result['predicted_price(x10k)'],key=lambda x:float(x))
```

```
Enter a number of Beds: 2
Enter a number of Baths: 2
Enter a number of sqft: 2432
   beds  baths  sqft  90042  91325  91326  91329  91331  91335  predicted_price(x10k)
1     2      2  2432      1      0      0      0      0      0              115.755043
2     2      2  2432      0      1      0      0      0      0               90.275177
3     2      2  2432      0      0      1      0      0      0               93.211914
4     2      2  2432      0      0      0      1      0      0               89.840073
5     2      2  2432      0      0      0      0      1      0               65.570496
6     2      2  2432      0      0      0      0      0      1               96.480980
```

```
print("For the given user input data:")
print("Bedrooms : " +ip_beds+ "  Bathroom's : " +ip_baths+ "  SQFT : " +ip_sqft)
print("The best recommended price is $" +str(res_min))
```

```
For the given user input data:
Bedrooms : 2  Bathroom's : 2  SQFT : 2432
The best recommended price is $65.57049560546875
```

Recommended the best minimum price in (10k) for house with the given requirements (bedrooms, bathrooms and square feet) along with the zip code.

- Clean-up Function

```
def refresh_database_environment():
  cnx = establish_database_connection()
  query = "DROP TABLE IF EXISTS house_details_results_preliminary;"
  cursor = cnx.cursor()
  cursor.execute(query)
  cnx.commit()
  cursor.close()
  cnx.close()

  cnx = establish_database_connection()
  query = "DROP TABLE IF EXISTS house_details_workers;"
  cursor = cnx.cursor()
  cursor.execute(query)
  cnx.commit()
  cursor.close()
  cnx.close()
  return
```

```
#disabled to avoid accidental drops
refresh_database_environment()
```

Implemented clean-up function to drop all the tables from the MySQL database environment.

Zillow App CSV files

Sheet 1: Home

	Unname	id	worker_id	Listed_Price	Address	Zipcode	Bedrooms	Bathrooms	SQFT	Status
0	0	1	2	$435,000.00	5059 1/2 Sh	90042	2	1	720	Condo for sale
1	1	2	2	$895,000.00	118 S Avenu	90042	2	3	1370	Townhouse for
2	2	3	2	$1,395,000.00	5055 Merid	90042	4	3	1968	House for sale
3	3	4	2	$599,000.00	258 S Avenu	90042	2	1	924	House for sale
4	4	5	2	$789,000.00	979 Dexter	90042	3	1	951	House for sale
5	5	6	2	$750,000.00	4225 Via Ar	90042	2	3	1682	Townhouse for
6	6	7	2	$630,000.00	4049 Via Ma	90042	2	2	1183	Condo for sale

Link	Description	Zestimate	Image	Full Details	Predicted_Price	Price Difference	Predicted_Price_(x10k)
https://www.zillow	Nestled on the	$0.00	https://photos.zill	2bds | 1 ba | 720 sqft	$689,394.60	$254,394.60	68.93946
https://www.zillow	NELA's (Northl	$880,347.00	https://maps.goog	2bds | 3 ba | 1370 sqft	$761,445.20	-$133,554.80	76.14452
https://www.zillow	Newly Remod	$0.00	https://photos.zill	4bds | 3 ba | 1968 sqft	$1,155,144.30	-$239,855.70	115.51443
https://www.zillow	For the Urbani	$614,164.00	https://photos.zill	2bds | 1 ba | 924 sqft	$762,432.30	$163,432.30	76.24323
https://www.zillow	Gorgeous, priv	$829,137.00	https://photos.zill	3bds | 1 ba | 951 sqft	$870,737.10	$81,737.10	87.07371
https://www.zillow	Birds-Eye View	$759,615.00	https://photos.zill	2bds | 3 ba | 1682 sqft	$885,762.70	$135,762.70	88.57627
https://www.zillow	Located in Mo	$555,811.00	https://photos.zill	2bds | 2 ba | 1183 sqft	$765,282.44	$135,282.44	76.528244

- Full details column displays the no. of bedrooms, bathrooms and sqft at a glance.
- Price difference is the difference between Predicted price and the listed price.

Zillow App CSV file

Sheet 2: Options

Purchase Type	Bedroom Data	Bedroom Displ	Bathroom Data	Bathroom Display
Coming soon	0	Any	0	Any
Condo for sale	1	1+	1	1+
House for sale	2	2+	2	2+
Lot / Land for sa	3	3+	3	3+
Multi-family hom	4	4+	4	4+
New constructio	5		5	
Townhouse for s	6		6	
Under Contract	7		7	
	12		11	

- The options sheet is used to provide filters on status of the house, number of bedrooms, and number of bathrooms.
- Users sheet stores users data (what filters they have applied to) and it can allow to access advanced features (save favorite houses).
- Contact us sheet is used to create a contact us page(filled with contact details of team members).

Sheet 3: Users

Email	First Name	Last Name	Status	Bedrooms	Bathrooms
pragatikoladiya@gmail.com	Pragati	Koladiya	House for sale	0	0

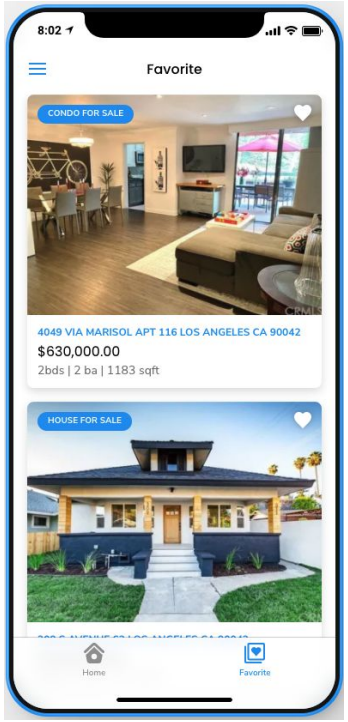
Sheet 4: Contact Us

Contact Name	Contact Details	Contact Number	Image
Pragati Koladiya	pragatidobariya10@gmail.com	4152591124	https://drive.google.com/file/d/1ng-EupG6NY42jHgbQNJGLWN9kEEYnbKH/view?usp=sharing
Priyanka Kanukollu	kmpriyanka492@gmail.com	-	https://drive.google.com/file/d/157Ue_WyBZ_8pTcYv5i2wZo7sr8FerkIG/view?usp=sharing
Tanvi Bhagat	tanvibhagat1005@gmail.com	-	https://drive.google.com/file/d/18zNZLXcEC6khQNUVfDMsJhqKMiFzdTMg/view?usp=sharing
Ketaki Joshi	ketakijoshi2502@gmail.com	-	

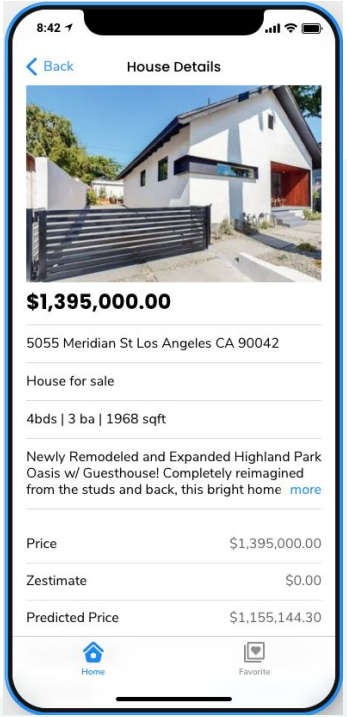
Welcome to Zillow App

Zillow App Download Link

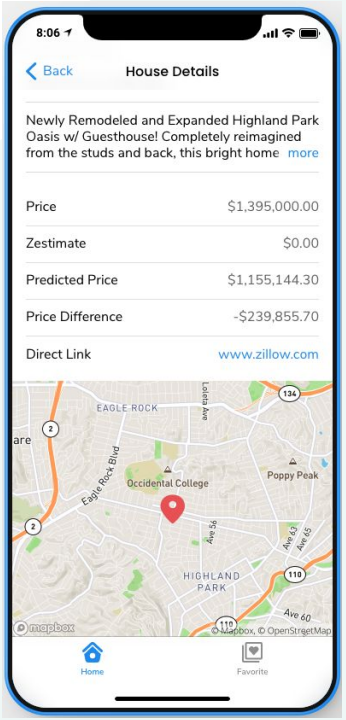
Home Page



Favorites Page

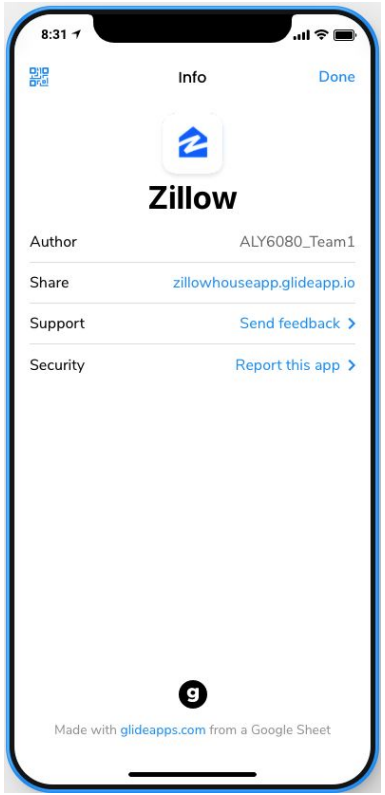


House Details Page

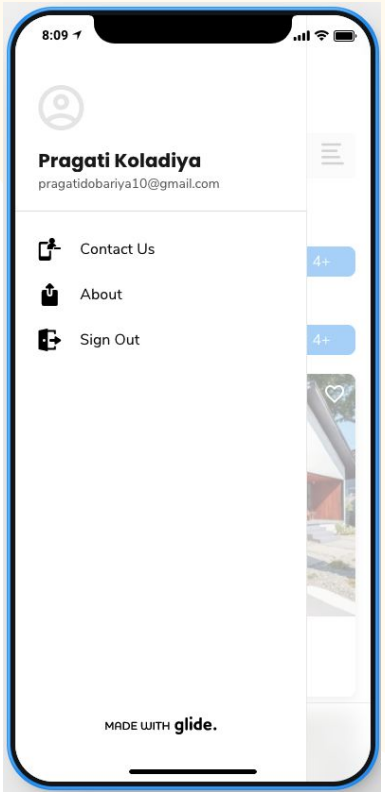


House Details Footer

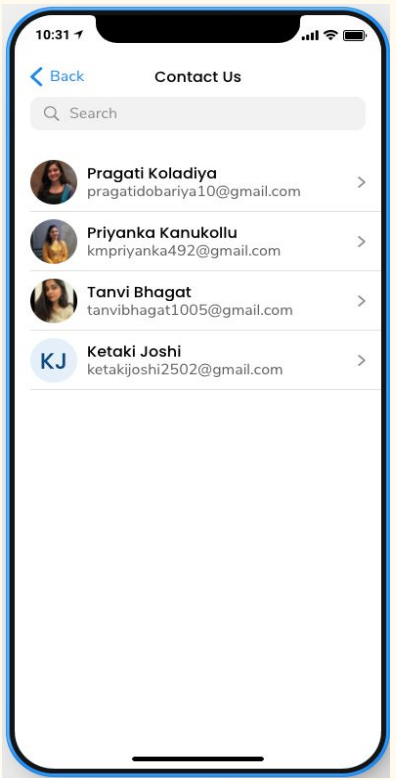
More pages on Zillow App



App Information



Side Menu



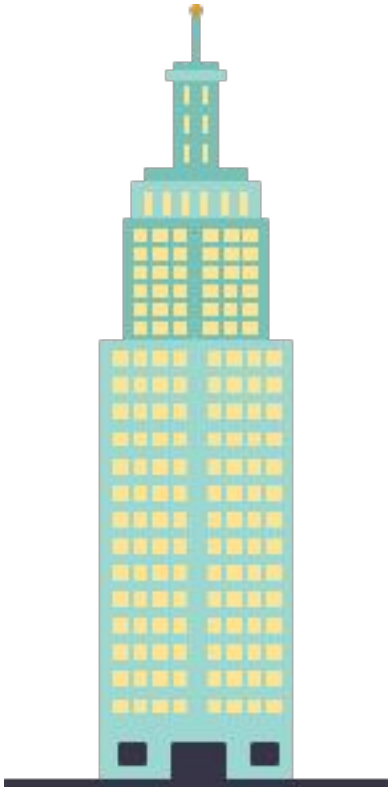
Contact Us Page

Links to the Code, App and Data

Python Code:
https://colab.research.google.com/drive/1GIy_PifYIHDZeF7GcQLTSyJIqxRjm6KX

Zillow App:
https://zillowhomes.glideapp.io/

Zillow Data:
https://docs.google.com/spreadsheets/d/1nXN1GQ_50EYCL2FXPHupRzJpb1BdBb4i8F_g8Ql9b5I/edit#gid=0



Conclusion

- Implemented Web scraping from Zillow website using Beautiful Soup and stored the data in MySQL database after removing HTML tags.
- Fetched the data for house price description, zestimate and image url.
- Implemented four different models for house price prediction using TensorFlow and found DNN regression with multiple variables to be the best model based on MAE.
- The built model can predict the house prices for the six mentioned zip codes and eventually recommend the lowest price amongst the predicted prices as the best recommended price for the customized user input.
- The scraped data is being used to develop the zillow app where users can have access to the houses which are for sale. The app will display few filters such as number of bedrooms, bathrooms and choose the status of the house on the basis of user preference. The option to save the favorite houses is only accessible to the signed up users.
- Each house displays the details such as actual price, zillow's estimated price, predicted price and price difference between actual and predicted price (using the best performing model).

References

- Zillow houses- https://www.zillow.com/
- House Price prediction- https://towardsdatascience.com/simple-house-price-predictor-using-ml-through-tensorflow-in-python-cbd2b637904b
- Web Scraping using beautiful soup- https://www.scrapingbee.com/blog/web-scraping-101-with-python/
- Tensor flow Regression Models-https://www.tensorflow.org/tutorials/keras/regression
- Build an App from a Google Sheet with Glide! https://www.youtube.com/watch?v=RQYxJvE-0MA&t=657s
- Lesson 19: How to use the in app filter, sort, group and advanced filtering in Glide https://www.youtube.com/watch?v=X0f2_JCa06Q

Thank You