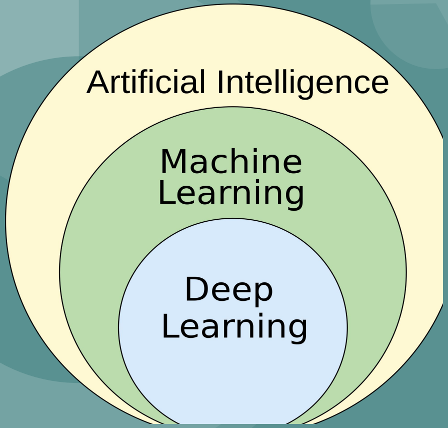
CS550 - Machine Learning and Business Intelligence



End-to-end Machine Learning project

By: Maranata Muluneh

Instructor: Dr. Chang, Henry

2023

Table of Content

Introduction

An **End-to-End Machine** Learning project involves building a complete pipeline that takes raw data as input, trains a machine learning model on that data, and then uses the trained model to make predictions on new, unseen data. This type of project typically includes several stages, such as data collection and preprocessing, feature engineering, model selection and training, and deployment of the trained model.

Introduction

The **goal of an End-to-End Machine** Learning project is to create a reliable and accurate model that can be used to solve a specific problem, such as image classification, natural language processing, or predictive analytics. This involves not only selecting the appropriate algorithms and techniques for the task at hand, but also understanding the domain-specific requirements and constraints, as well as optimizing the model's performance.

Theory

Basic steps in an End to End Machine learning project includes the following:

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Theory

Step 1: Look at the big picture

In this step we will have the following steps

- Framing the problem
- Selecting a performance measure
- Checking the assumptions

Theory

Step 2: Get the data.

After figuring out the problem and visioning the approach to solve the specific problem we will have the following steps

- Creating the workspace
- Downloading the data and
- Creating a test set

Theory

Step 3: Discover and visualize the data to gain insights.

To gain more insight of the data we can visualize the data by:

- Visualizing the data
- Looking for correlations
- Experimenting with attribute combinations

Theory

Step 4: Prepare the data for Machine Learning algorithms.

This the first technical step of creating the model which is the preprocessing model where we clean and prepare the data to create the model

- Data cleaning
- Handling text and categorical attributes
- Using custom transformers
- Featuring scaling
- Using transformation pipelines

Theory

Step 5: Select a model and train it.

Includes the steps below:

- Training and evaluating on the training set
- Better evaluation using the cross validation parameter vs Hyperparameter

Theory

Step 6: Fine-tune your model.

This the last step of creating the model, includes:

- Analyzing the best models and their errors
- Evaluating your system on the test set

Step 7:Present your solution.

Step 8: Launch, monitor, and maintain your system.

Implementation (Using Colab)

Environment : Colab, Tensorflow 2

Programming Language: Python

Libraries

(Common Libraries)

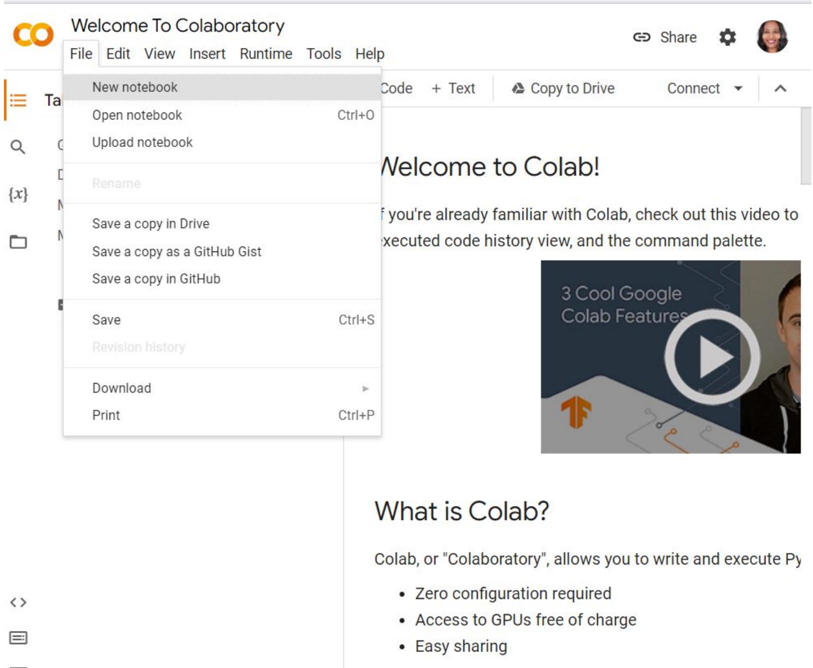
```
import sys
import numpy as np
import pandas as pd

import os
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Implementation

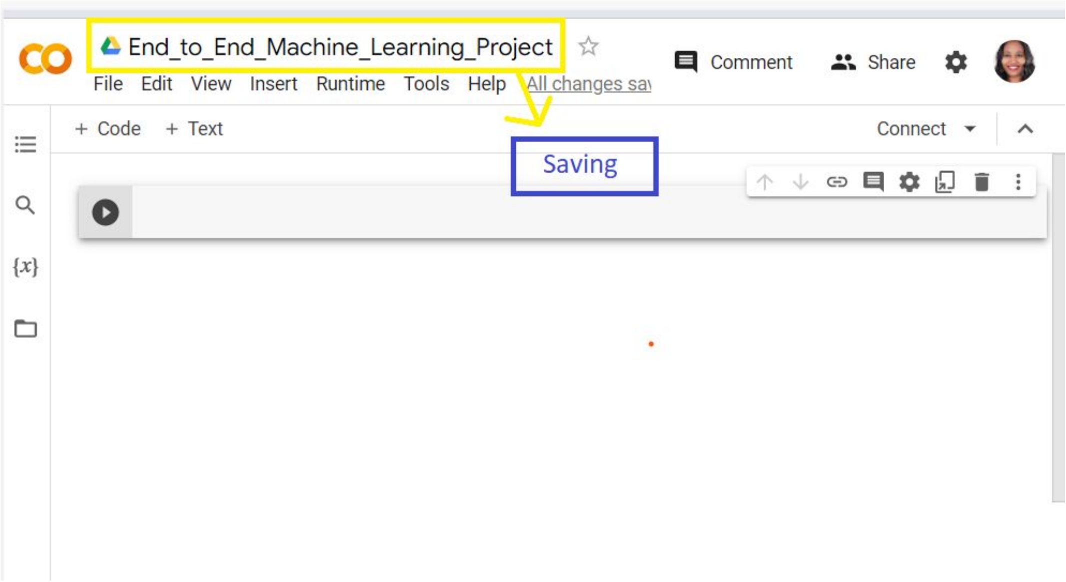
- Here's some implementation of End to End Machine Learning Project using Colab:

Step 1: Setting up our Colab



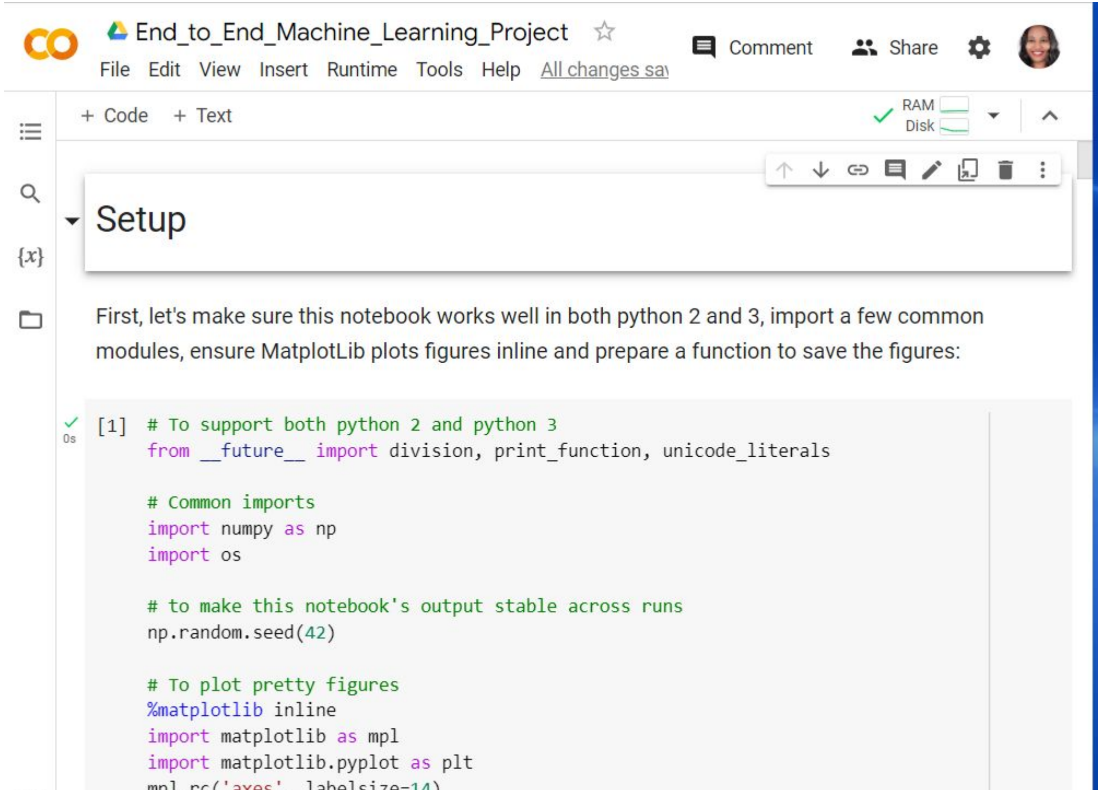
Implementation

Step 1: Setting up our Colab



Implementation

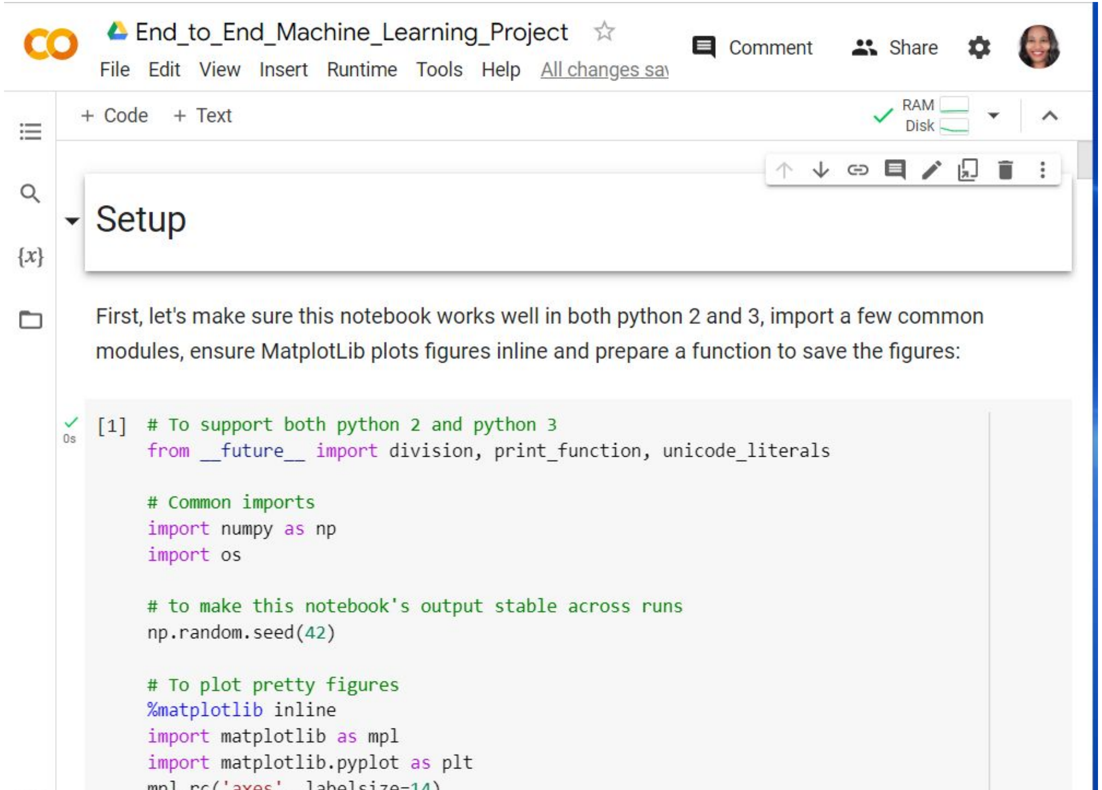
Step 2: Setting up the Colab notebook for the desired version (python 2 or 3) by importing some modules



Implementation

Step 3: Getting the data

We have housing data residing in California that contains properties of a house like a longitude,latitude,housing_median_age,total_rooms,total_bedrooms, population,households, median_income, median_house_value and ocean_proximit



Implementation

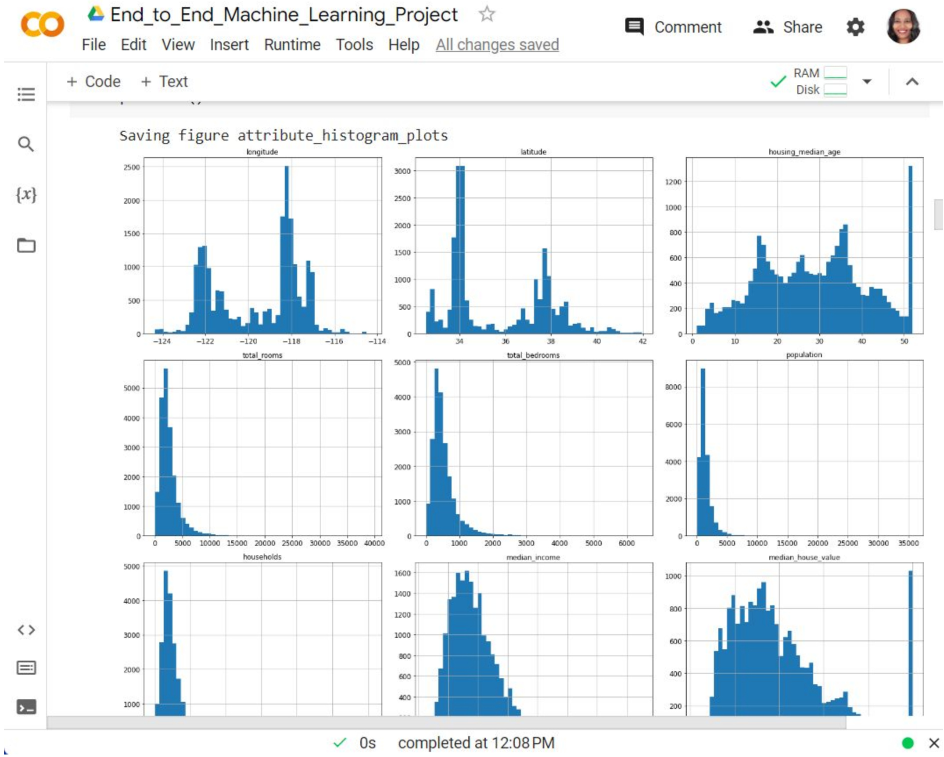
After Fetching the data, we can have the loaded data on our Colab, as shown below.

```
housing = load_housing_data()
housing.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

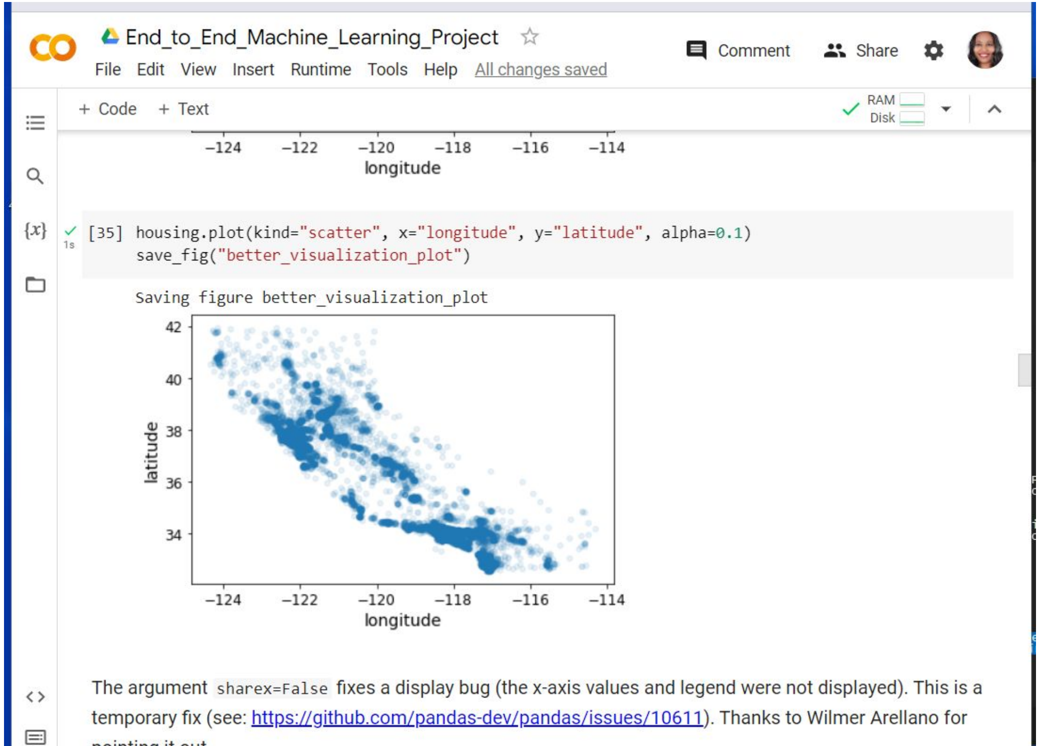
Implementation

We have the generated figure for the histogram information of the house.

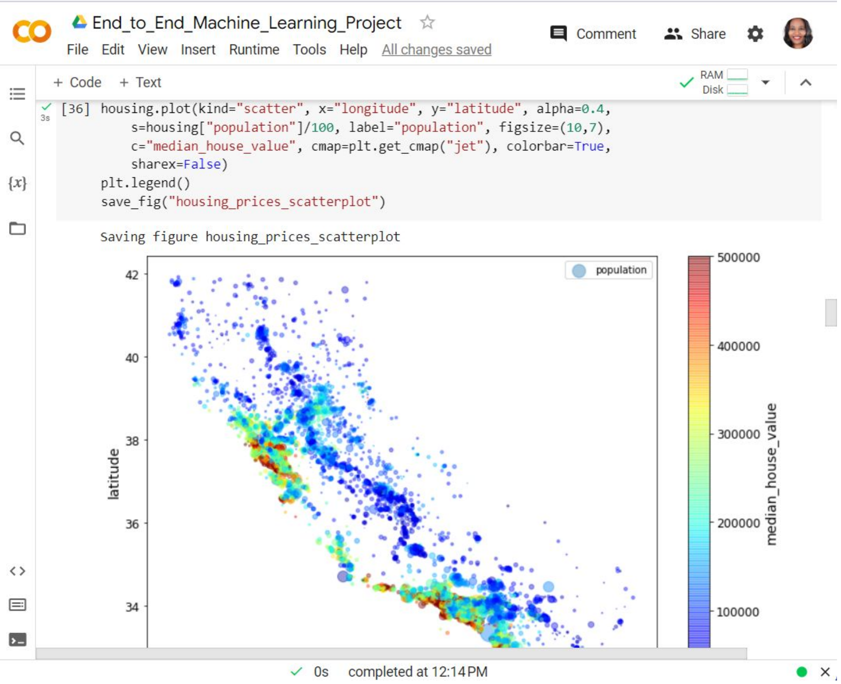


Implementation

Step 4: Discover and visualize the data to gain insights



Implementation



Implementation

Step 5: Prepare the data for Machine Learning algorithms

- While cleaning the data, we have the preprocessing phase, where we clean the data before using the data to create the machine learning model.
- The preprocessing steps include cleaning the data that is filling in the missing information or dropping unnecessary columns or rows that are not important later for the model to be created, transformation, and other steps essential to make the model.

Implementation

Step 5: Prepare the data for Machine Learning algorithms

- While cleaning the data, we have the preprocessing phase, where we clean the data before using the data to create the machine learning model.
- The preprocessing steps include cleaning the data that is filling in the missing information or dropping unnecessary columns or rows that are not important later for the model to be created, transformation, and other steps essential to make the model.

Implementation

End_to_End_Machine_Learning_Project

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text RAM Disk

Prepare the data for Machine Learning algorithms

```
[47] housing = strat_train_set.drop("median_house_value", axis=1) # drop labels for training set
     housing_labels = strat_train_set["median_house_value"].copy()
```

```
[48] sample_incomplete_rows = housing[housing.isnull().any(axis=1)].head()
     sample_incomplete_rows
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househol
1606	-122.08	37.88	26.0	2947.0	NaN	825.0	626
10915	-117.87	33.73	45.0	2264.0	NaN	1970.0	499
19150	-122.70	38.35	14.0	2313.0	NaN	954.0	397
4186	-118.23	34.13	48.0	1308.0	NaN	835.0	294
16885	-122.40	37.58	26.0	3281.0	NaN	1145.0	480

```
[49] sample_incomplete_rows.dropna(subset=["total_bedrooms"])    # option 1
```

0s completed at 12:29 PM

End_to_End_Machine_Learning_Project

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text RAM Disk

```
[49] sample_incomplete_rows.dropna(subset=["total_bedrooms"])    # option 1
```

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households

```
[50] sample_incomplete_rows.drop("total_bedrooms", axis=1)       # option 2
```

	longitude	latitude	housing_median_age	total_rooms	population	households	median_inc
1606	-122.08	37.88	26.0	2947.0	825.0	626.0	2.9
10915	-117.87	33.73	45.0	2264.0	1970.0	499.0	3.4
19150	-122.70	38.35	14.0	2313.0	954.0	397.0	3.7
4186	-118.23	34.13	48.0	1308.0	835.0	294.0	4.2
16885	-122.40	37.58	26.0	3281.0	1145.0	480.0	6.3

```
[51] median = housing["total_bedrooms"].median()
     sample_incomplete_rows["total_bedrooms"].fillna(median, inplace=True) # option 3
     sample_incomplete_rows
```

0s completed at 12:29 PM

Implementation

Step 6: Select and train a model

End_to_End_Machine_Learning_Project

File Edit View Insert Runtime Tools Help All changes saved

Select and train a model

```
[85] from sklearn.linear_model import LinearRegression

     lin_reg = LinearRegression()
     lin_reg.fit(housing_prepared, housing_labels)

LinearRegression()
```

```
[86] # let's try the full preprocessing pipeline on a few training instances
     some_data = housing.iloc[:5]
     some_labels = housing_labels.iloc[:5]
     some_data_prepared = full_pipeline.transform(some_data)

     print("Predictions:", lin_reg.predict(some_data_prepared))

Predictions: [ 85657.90192014 305492.60737488 152056.46122456 186095.70946094
 244550.67966089]
```

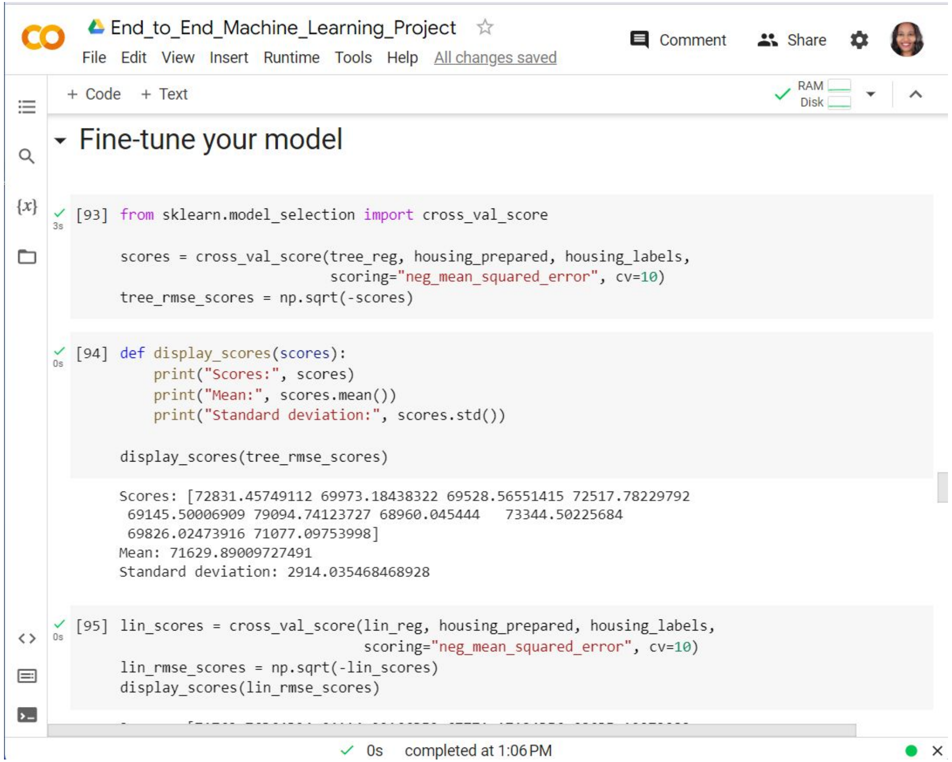
Compare against the actual values:

```
[87] print("Labels:", list(some_labels))

Labels: [72100.0, 279600.0, 82700.0, 112500.0, 238300.0]
```

Implementation

Step 7: Fine-tune your model



Implementation

End_to_End_Machine_Learning_Project

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

The best hyperparameter combination found:

```
[102] grid_search.best_params_
```

```
{'max_features': 8, 'n_estimators': 30}
```

```
[103] grid_search.best_estimator_
```

```
RandomForestRegressor(max_features=8, n_estimators=30, random_state=42)
```

Let's look at the score of each hyperparameter combination tested during the grid search:

```
[104] cvres = grid_search.cv_results_
      for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
          print(np.sqrt(-mean_score), params)
```

```
63895.161577951665 {'max_features': 2, 'n_estimators': 3}
54916.32386349543 {'max_features': 2, 'n_estimators': 10}
52885.86715332332 {'max_features': 2, 'n_estimators': 30}
60075.3680329983 {'max_features': 4, 'n_estimators': 3}
52495.01284985185 {'max_features': 4, 'n_estimators': 10}
50187.24324926565 {'max_features': 4, 'n_estimators': 30}
58064.73529982314 {'max_features': 6, 'n_estimators': 3}
51519.32062366315 {'max_features': 6, 'n_estimators': 10}
49969.80441627874 {'max_features': 6, 'n_estimators': 30}
58895.824998155826 {'max_features': 8, 'n_estimators': 3}
52459.79624724529 {'max_features': 8, 'n_estimators': 10}
```

✓ 0s completed at 1:06 PM

End_to_End_Machine_Learning_Project

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk

```
[111] final_model = grid_search.best_estimator_

      X_test = strat_test_set.drop("median_house_value", axis=1)
      y_test = strat_test_set["median_house_value"].copy()

      X_test_prepared = full_pipeline.transform(X_test)
      final_predictions = final_model.predict(X_test_prepared)

      final_mse = mean_squared_error(y_test, final_predictions)
      final_rmse = np.sqrt(final_mse)
```

```
final_rmse
```

```
47873.26095812988
```

We can compute a 95% confidence interval for the test RMSE:

```
[113] from scipy import stats
```

```
[114] confidence = 0.95
      squared_errors = (final_predictions - y_test) ** 2
      mean = squared_errors.mean()
      m = len(squared_errors)

      np.sqrt(stats.t.interval(confidence, m - 1,
                               loc=np.mean(squared_errors),
```

✓ 0s completed at 1:06 PM

Conclusion

In conclusion, successful End-to-End Machine Learning projects require a combination of technical expertise, domain knowledge, and project management skills. They involve collaboration between data scientists, software engineers, domain experts, and stakeholders to ensure that the project meets its objectives and delivers value to the business or organization. Overall, End-to-End Machine Learning projects can be challenging but rewarding endeavors that enable organizations to harness the power of machine learning to gain insights, make predictions, and drive innovation.

References

- Ageron. (2021, October 11). Handson-ML2/02_end_to_end_machine_learning_project.ipynb at master · Ageron/Handson-ML2. GitHub. Retrieved February 20, 2023, from https://github.com/ageron/handson-ml2/blob/master/02_end_to_end_machine_learning_project.ipynb
- profile.php?id=100001802069241. (2021, August 19). How to develop an end-to-end machine learning project and deploy it to Heroku with flask. freeCodeCamp.org. Retrieved February 20, 2023, from https://www.freecodecamp.org/news/end-to-end-machine-learning-project-turorial/