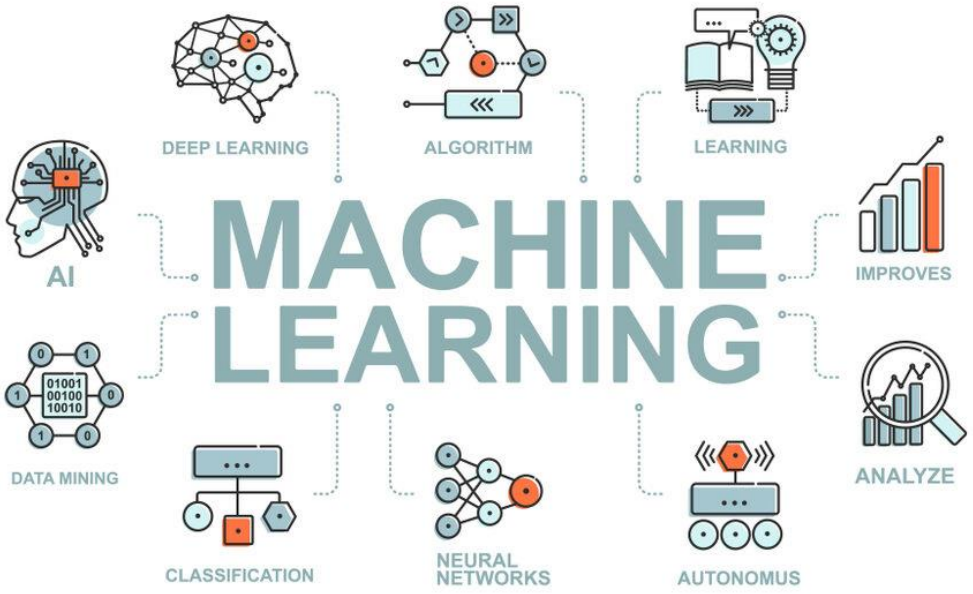


INDUSTRIAL TRAINING

MACHINE LEARNING

by

Internshala Training

MACHINE LEARNING BY INTERNSHALA TRAINING

A SUMMER TRAINING REPORT

Submitted by:

AMANJOT SINGH

20BCS6702

in partial fulfillment of summer training for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

JULY 2022

About the Company

Company Name: **Internshala Trainings**

Company's website: internshala.com

Internshala is an internship and online training platform, based in Gurgaon, India. Founded by Sarvesh Agrawal, an IIT Madras alumnus, in 2011, the website helps students find internships with organizations in India.

Internshala is India's no.1 internship and training platform with 40000+ paid internships in Engineering, MBA, media, law, arts, and other streams.

Company's Vision:

Internshala is a tech company on a mission to equip students with relevant skills & practical exposure to help them get the best possible start in their careers. Imagine a world full of freedom and possibilities. A world where you can discover your passion and turn it into your career. A world where you graduate fully assured, confident, and prepared to stake a claim on your place in the world.

CERTIFICATE

INTERNSHALA TRAININGS

Certificate of Training

Amanjot Singh

from Chandigarh University has successfully completed a 6-week online training on **Machine Learning**. The training consisted of Introduction to Machine Learning, Data, Introduction to Python, Data Exploration and Pre-processing, Linear Regression, Introduction to Dimensionality Reduction, Logistic Regression, Decision Tree, Ensemble Models, and Clustering (Unsupervised Learning) modules.
In the final assessment, Amanjot scored 68% marks.
We wish Amanjot all the best for future endeavours.

Sarvesh Agarwal
FOUNDER & CEO, INTERNSHALA

Date of certification: 2022-07-31 Certificate no. : 77CBF843-F587-BB66-3F32-5768FABBF6AA
For certificate authentication, please visit https://trainings.internshala.com/verify_certificate

https://trainings.internshala.com/s/v/1788243/75b38894

ACKNOWLEDGEMENT

I, Amanjot Singh acknowledge the Internshala Training platform for providing the best machine learning training courses with doubt-clearing support.

ABSTRACT

In the training session of Industrial training, I learned about Introduction to Machine Learning to data cleaning followed by training and prediction which includes the following:

- Introduction to Machine Learning
- Data
- Introduction to Python
- Data Exploration and Pre-processing
- Linear Regression
- Introduction to Dimensionality reduction
- Logistic Regression
- Decision Tree
- Ensemble Models
- Clustering

Table of Content

Title Page	2
About the Company	3
Certificate	4
Acknowledgment	5
Abstract	5
Table of content	6
Introduction	7
Theory	10
Projects/Tasks	13
Methodology Adopted	32
Results and Discussion	34
Conclusion	34

INTRODUCTION

Machine Learning: Machine learning is the ability of machines, that is, computers to learn and improve from their past experience or data, without being explicitly programmed.

Applications of ML:

- Facebook newsfeed,
- Facebook photo auto-tagging feature,
- Product recommendations by shopping portals.

Advanced applications of ML:

- Identifying frauds in banking,
- Sentiment analysis,
- Amazon go,
- Chatbots,
- Self-driven cars.

Types of ML:

1. Supervised Learning
2. Unsupervised Learning

- Labeled data: The data which contains a target variable or an output variable that answers a question of interest is called labeled data.

- Unlabeled Data: The data which contains information about something but does not have a predefined target variable.

Supervised Learning Models:

1. Classification Models: Target - Categorical variable
2. Regression Models: Target - Continuous numerical variable

- Unsupervised Learning: The machine learning that is deployed to find patterns in unlabelled data is referred to as unsupervised machine learning.

- Variable: A variable represents one specific characteristic of the data or tells one specific information about the data under consideration.

Types of variables:

Numeric Variable - those which are quantifiable

1. Continuous variables – are those numeric variables that can take any value between a certain set of real numbers.
2. Discrete variables - those numeric variables which are countable and can take only whole numbers (i.e. integers) as value.

Categorical variable - those which are like adjectives and express a feeling or a characteristic, the categorical variable consists of string or text values.

1. Ordinal variables –those categorical variables that can be arranged in a logical order.
2. Nominal variables – are those categorical variables that cannot be ranked based on their values.

- Dependent variables are the ones whose values depend on other independent variables and cannot be changed easily.

- Independent variables are the ones whose values don't depend on any other variables.

- Structured data: Structured data always have a format of structure when they are stored.

- Unstructured data: Data that does not have a well-defined structure and is not arranged in any tabular format.

DATA SPLITTING

Train dataset:

- To educate or train our models
- The data on which a model is built is called training data

- Training data is used by the model to learn

Test dataset:

- To examine the model performance
- Once the model is trained, it is examined as to how well it has learned using another subset of the original data which is called Test Data
- The model predicts the target variable values for the test data and the predicted values are compared with actual values and checked as to how many of them were correctly predicted

FEATURE SCALING

- Feature Scaling is all about scaling the feature variables (i.e. all the independent variables) into the same range
- The variables are scaled to have similar magnitude and ranges so that model is not biased towards a particular variable
- Feature scaling is a must for those algorithms where some measures of distance between data points are involved like Logistic Regression, Linear Regression, K Nearest neighbors, Principal Components analysis, etc.
- However, Feature scaling is not required for tree-based algorithms like Random Forest, Decision Tree, etc.

THEORY

The Following topics and sub-topics have been thought during the summer training:

Introduction to Machine Learning

What is Machine Learning

How Machine learning works

Types of machine learning-Supervised Learning and Unsupervised learning

Data

Types of data

Graphical and Analytical Representation of data

Limitations of traditional data Analysis

Introduction To Python and Jupyter Notebook

Python

Jupyter Notebook

Google collaboratory

Basic libraries in python

Understanding the basics of python programming

Basic data Exploration

Advanced functions for data manipulation

Data Exploration and Pre-processing

Context Setting and Problem

Data exploration

Target variable

Independent variable

Categorical variable

Splitting of data

Feature Scaling of data

Linear Regression

Building the first predictive model with a mean prediction

Introduction to linear regression

Understanding gradient decent

Assumption of linear regression

Implementing linear regression

Feature engineering

Introduction to dimensionality reduction technique

Common dimensionality reduction techniques

Advanced dimensionality reduction techniques

Logistic Regression

Understanding the basics of logistic regression

Evaluation metrics

Implementing logistic regression

Decision tree

Introduction to Decision tree

The logic behind the decision tree

Implementing decision tree

Improving model performance by Pruning/Hyperparameters tuning

Ensemble Models

Basics of Ensemble Techniques

Random Forest

Implementing Bagging and Random Forest

Clustering (Unsupervised Learning)

Introduction to clustering

Evaluating clustering models

K-Means

Challenges in implementing K-Means

Implementation of K-Means

PROJECTS/TASKS

1. Breast Cancer Prediction System

Breast cancer is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases and affected over 2.1 Million people in 2015 alone. It starts when cells in the breast begin to grow out of control. These cells usually form tumors that can be seen via X-ray or felt as lumps in the breast area.

The key challenge against its detection is how to classify tumors into malignant (cancerous) or benign(noncancerous).

Data Set Information:
Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe the characteristics of the cell nuclei.

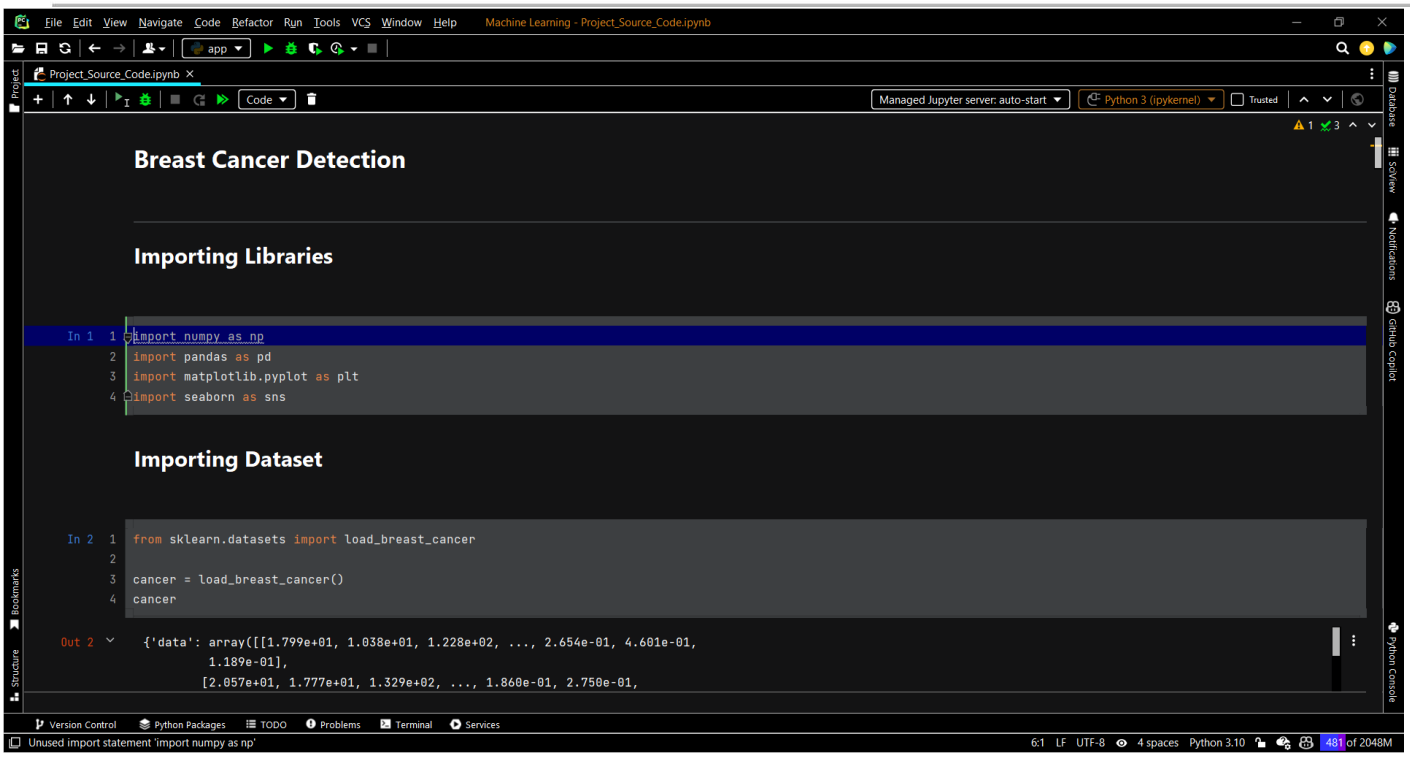
Attribute Information:

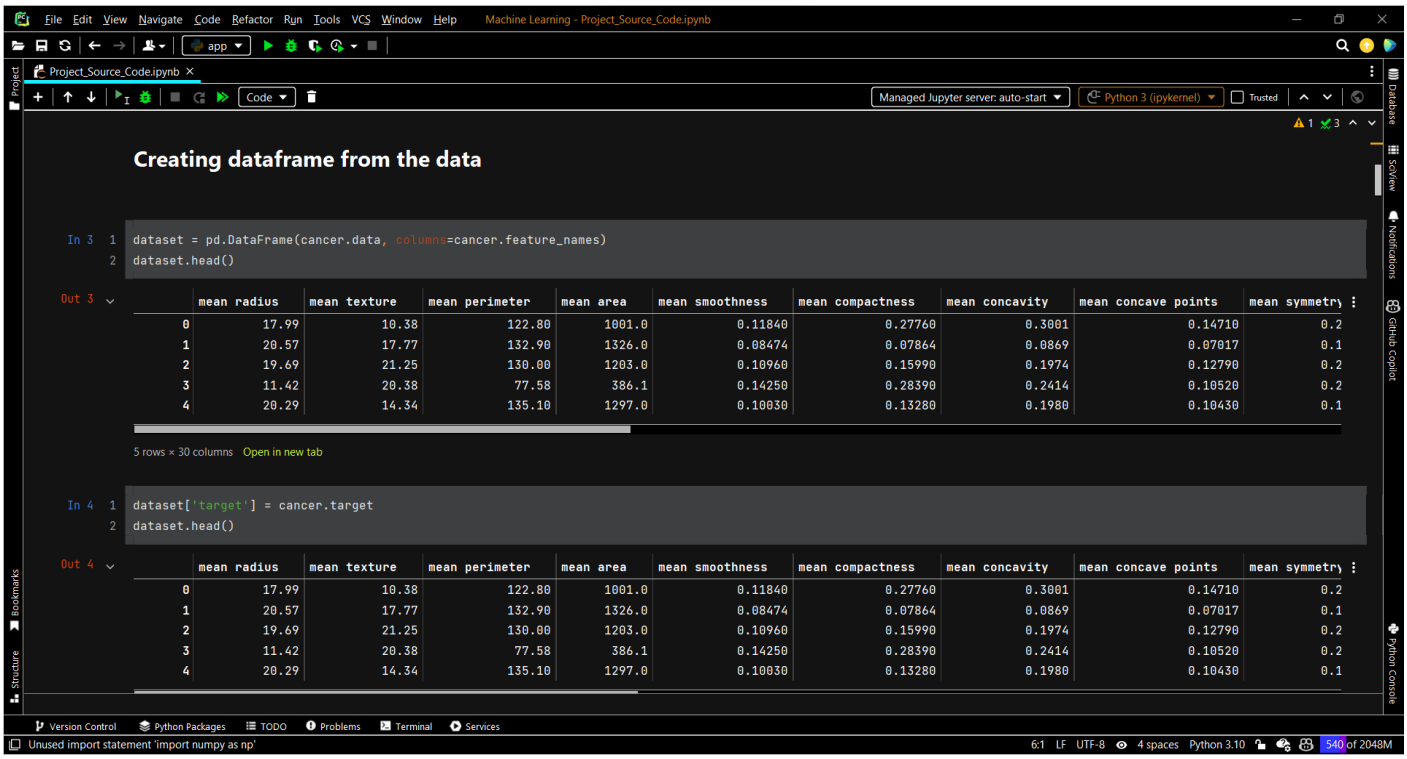
- ID number
- Diagnosis (M = malignant, B = benign)
- Ten real-valued features are computed for each cell nucleus (3–32):

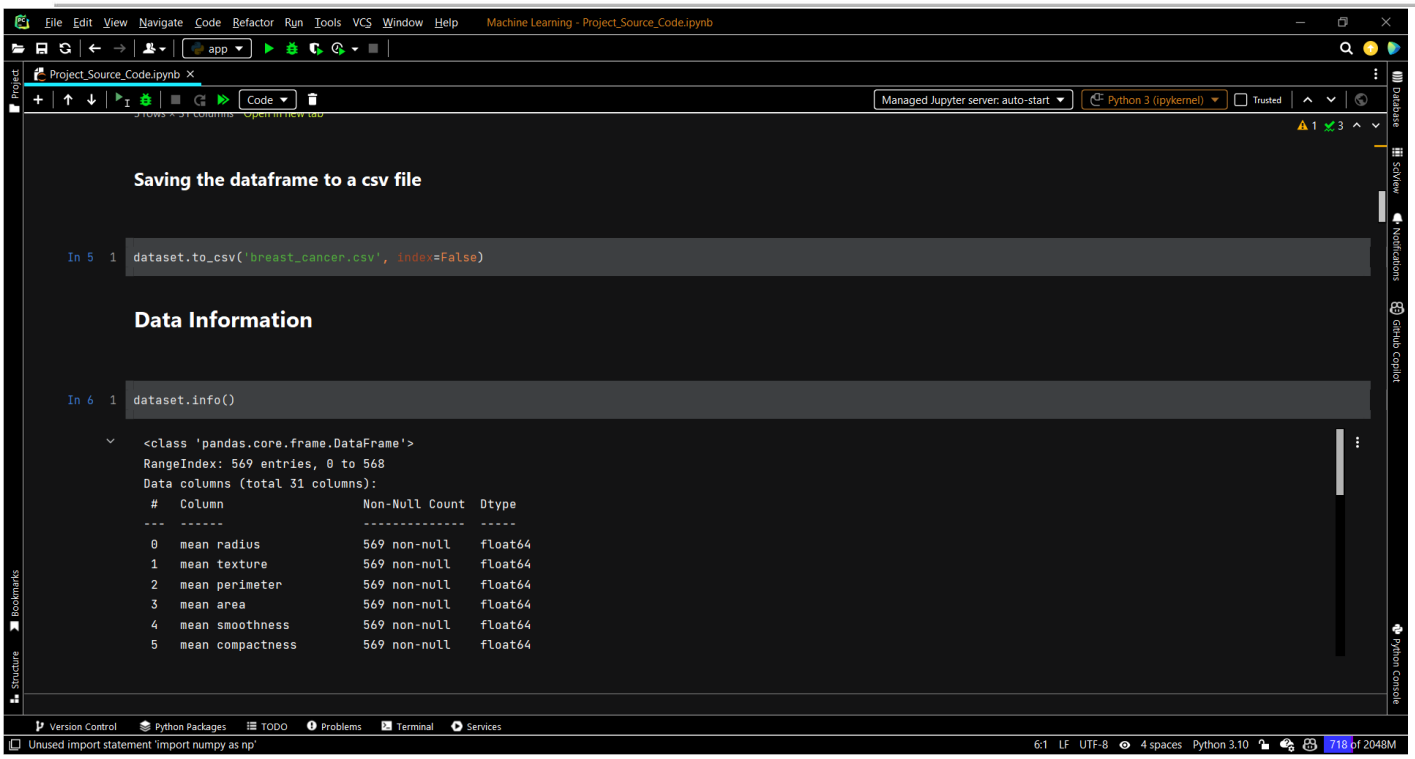
    a) radius (mean of distances from the center to points on the perimeter)
    b) texture (standard deviation of gray-scale values)
    c) perimeter
    d) area
    e) smoothness (local variation in radius lengths)
    f) compactness (perimeter² / area — 1.0)
    g) concavity (severity of concave portions of the contour)
    h) concave points (number of concave portions of the contour)
    i) symmetry
    j) fractal dimension (“coastline approximation” — 1)

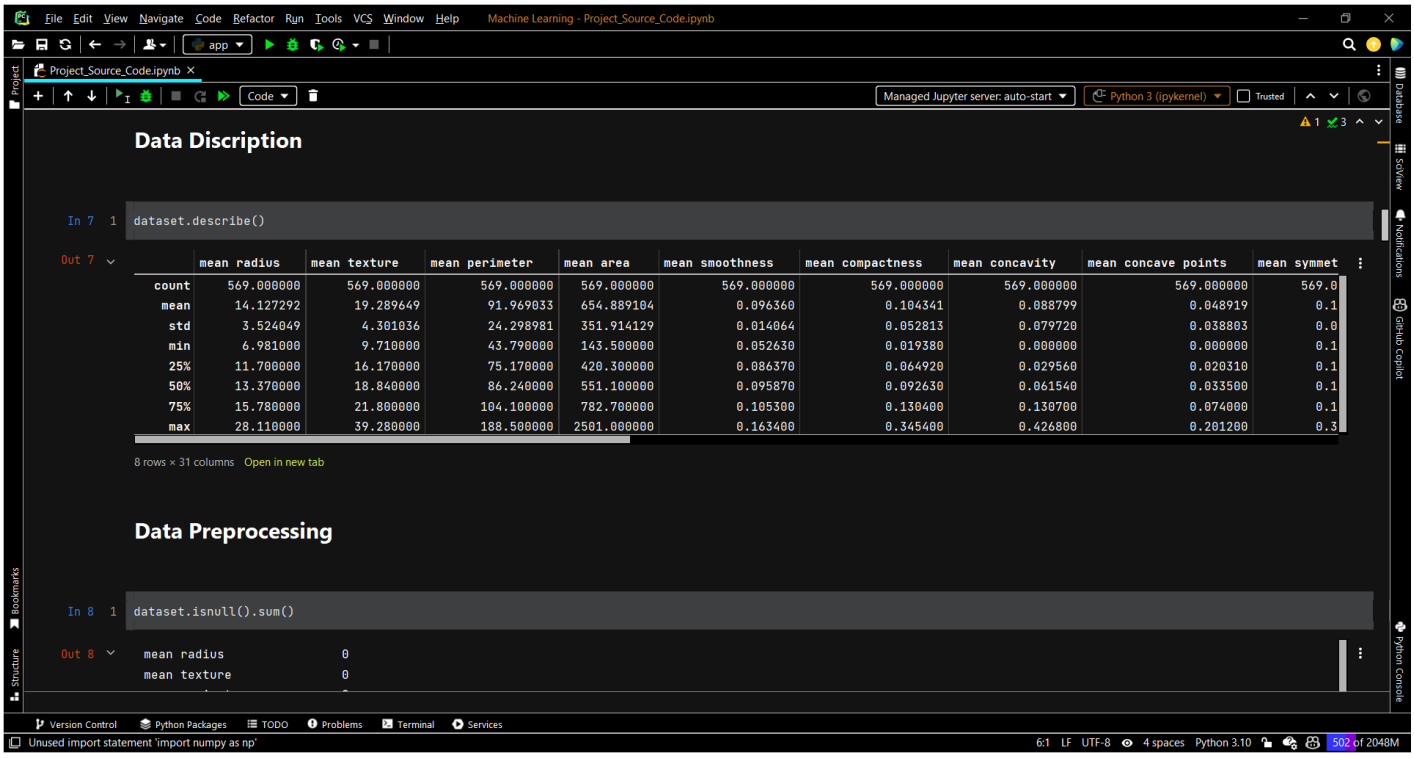
Github file link:- https://github.com/RITURAJRAMAN/Breast_Cancer_Detection_Project

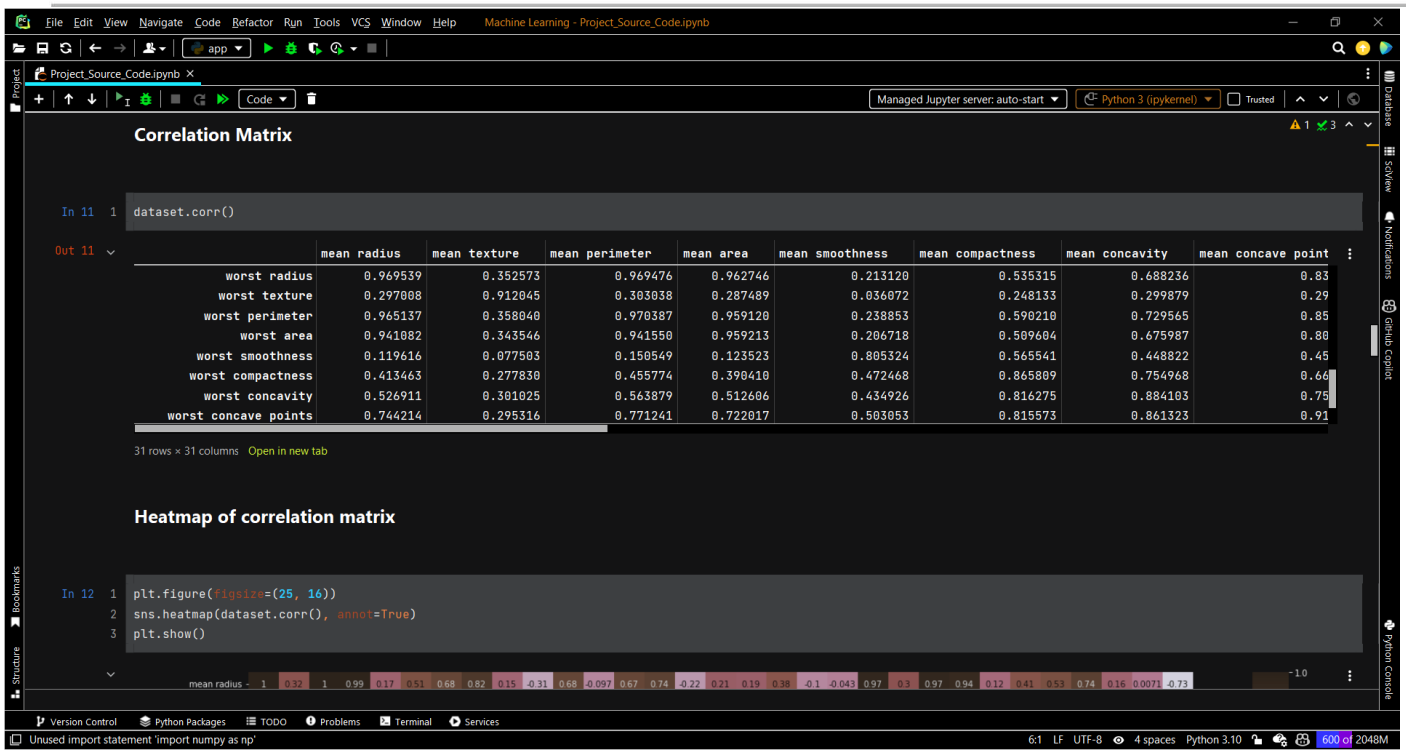
Live Breast cancer predictor web app:- https://breastcancer-predictor.azurewebsites.net/

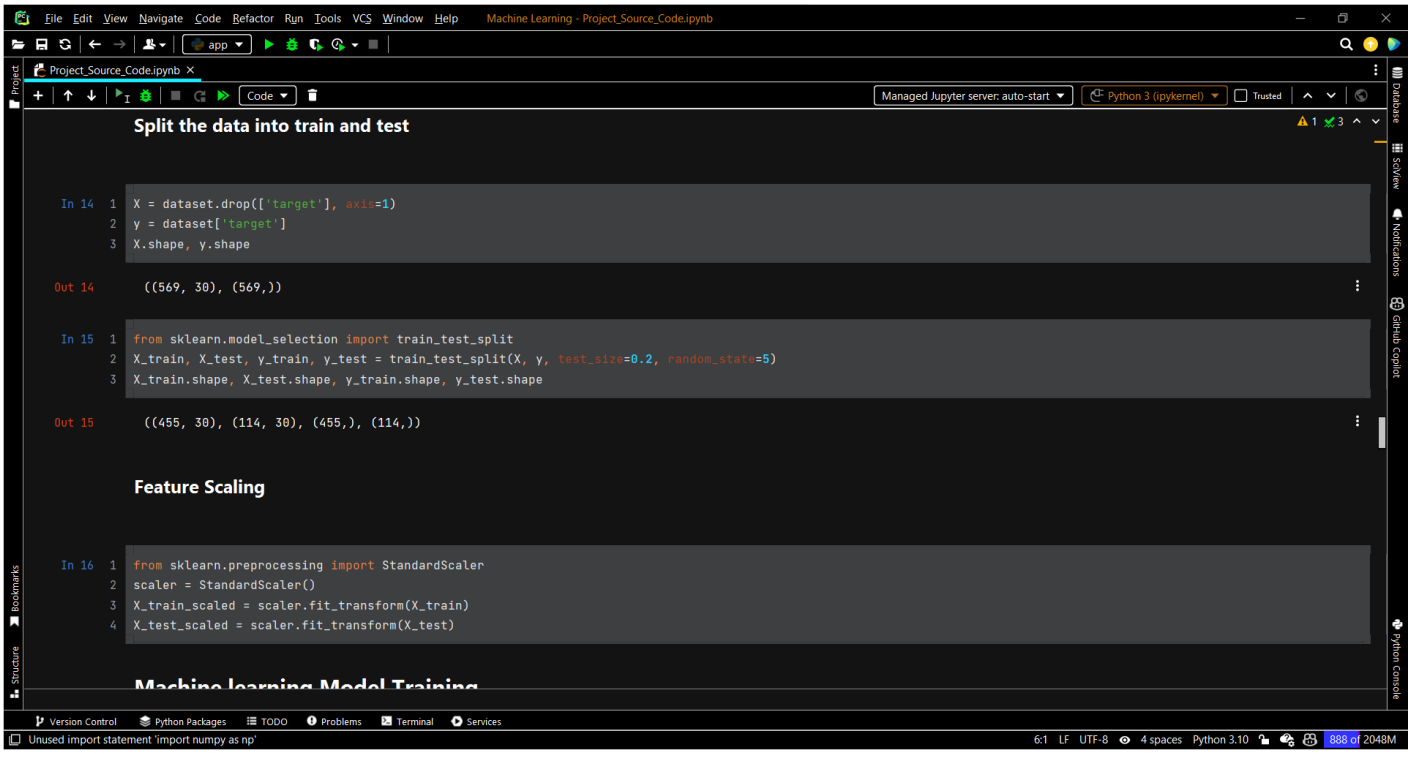

Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Breast Cancer Detection
Importing Libraries
In 1
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
Importing Dataset
In 2
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer
Out 2
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
1.189e-01],
[2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
Unused import statement 'import numpy as np'

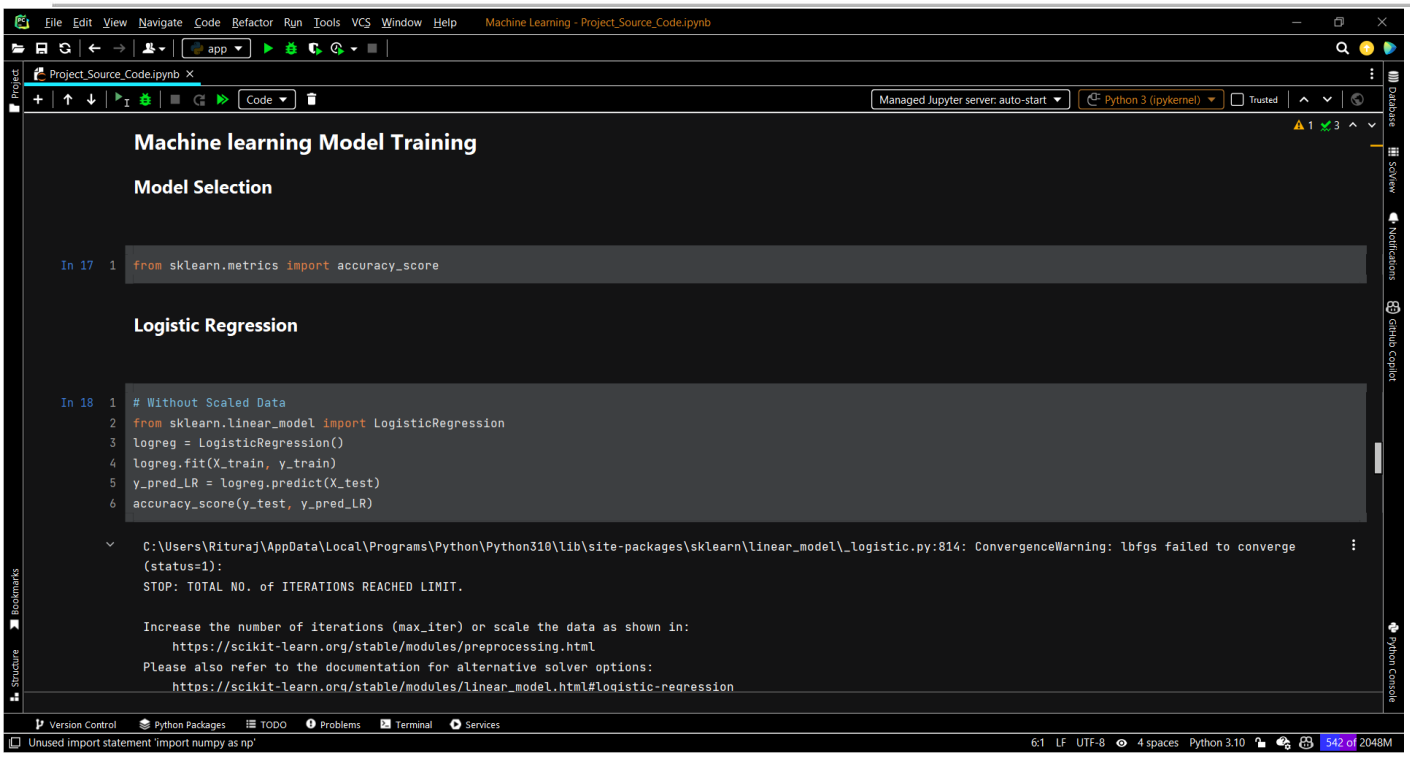

Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Creating dataframe from the data
In 3
dataset = pd.DataFrame(cancer.data, columns=cancer.feature_names)
dataset.head()
Out 3
mean radius mean texture mean perimeter mean area mean smoothness mean compactness mean concavity mean concave points mean symmetry
0 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2
1 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1
2 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2
3 11.42 20.38 77.58 386.1 0.14250 0.28390 0.2414 0.10520 0.2
4 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1
5 rows × 30 columns Open in new tab
In 4
dataset['target'] = cancer.target
dataset.head()
Out 4
mean radius mean texture mean perimeter mean area mean smoothness mean compactness mean concavity mean concave points mean symmetry
0 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710 0.2
1 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017 0.1
2 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790 0.2
3 11.42 20.38 77.58 386.1 0.14250 0.28390 0.2414 0.10520 0.2
4 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430 0.1
Unused import statement 'import numpy as np'


Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Saving the dataframe to a csv file
In 5
dataset.to_csv('breast_cancer.csv', index=False)
Data Information
In 6
dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 mean radius 569 non-null float64
1 mean texture 569 non-null float64
2 mean perimeter 569 non-null float64
3 mean area 569 non-null float64
4 mean smoothness 569 non-null float64
5 mean compactness 569 non-null float64
Unused import statement 'import numpy as np'


Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Data Discription
In 7
dataset.describe()
Out 7
mean radius mean texture mean perimeter mean area mean smoothness mean compactness mean concavity mean concave points mean symmet
count 569.000000 569.000000 569.000000 569.000000 569.000000 569.000000 569.000000 569.000000 569.0
mean 14.127292 19.289649 91.969033 654.889104 0.096360 0.104341 0.088799 0.048919 0.1
std 3.524049 4.301036 24.298981 351.914129 0.014064 0.052813 0.079720 0.038803 0.0
min 6.981000 9.710000 43.790000 143.500000 0.052630 0.019380 0.000000 0.000000 0.1
25% 11.700000 16.170000 75.170000 420.300000 0.086370 0.064920 0.029560 0.020310 0.1
50% 13.370000 18.840000 86.240000 551.100000 0.095870 0.092630 0.061540 0.033500 0.1
75% 15.780000 21.800000 104.100000 782.700000 0.105300 0.130400 0.130700 0.074000 0.1
max 28.110000 39.280000 188.500000 2501.000000 0.163400 0.345400 0.426800 0.201200 0.3
8 rows × 31 columns Open in new tab
Data Preprocessing
In 8
dataset.isnull().sum()
Out 8
mean radius 0
mean texture 0
Unused import statement 'import numpy as np'


Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Correlation Matrix
In 11
dataset.corr()
Out 11
mean radius mean texture mean perimeter mean area mean smoothness mean compactness mean concavity mean concave point
worst radius 0.969539 0.352573 0.969476 0.962746 0.213120 0.535315 0.688236 0.83
worst texture 0.297008 0.912045 0.303038 0.287489 0.036072 0.248133 0.299879 0.29
worst perimeter 0.965137 0.358040 0.970387 0.959120 0.238853 0.590210 0.729565 0.85
worst area 0.941082 0.343546 0.941550 0.959213 0.206718 0.509604 0.675987 0.80
worst smoothness 0.119616 0.077503 0.150549 0.123523 0.805324 0.565541 0.448822 0.45
worst compactness 0.413463 0.277830 0.455774 0.390410 0.472468 0.865809 0.754968 0.66
worst concavity 0.526911 0.301025 0.563879 0.512606 0.434926 0.816275 0.884103 0.75
worst concave points 0.744214 0.295316 0.771241 0.722017 0.503053 0.815573 0.861323 0.91
31 rows × 31 columns Open in new tab
Heatmap of correlation matrix
In 12
plt.figure(figsize=(25, 16))
sns.heatmap(dataset.corr(), annot=True)
plt.show()
Unused import statement 'import numpy as np'


Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Split the data into train and test
In 14
X = dataset.drop(['target'], axis=1)
y = dataset['target']
X.shape, y.shape
Out 14
((569, 30), (569,))
In 15
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
Out 15
((455, 30), (114, 30), (455,), (114,))
Feature Scaling
In 16
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
Machine learning Model Training
Unused import statement 'import numpy as np'



Machine learning Model Training

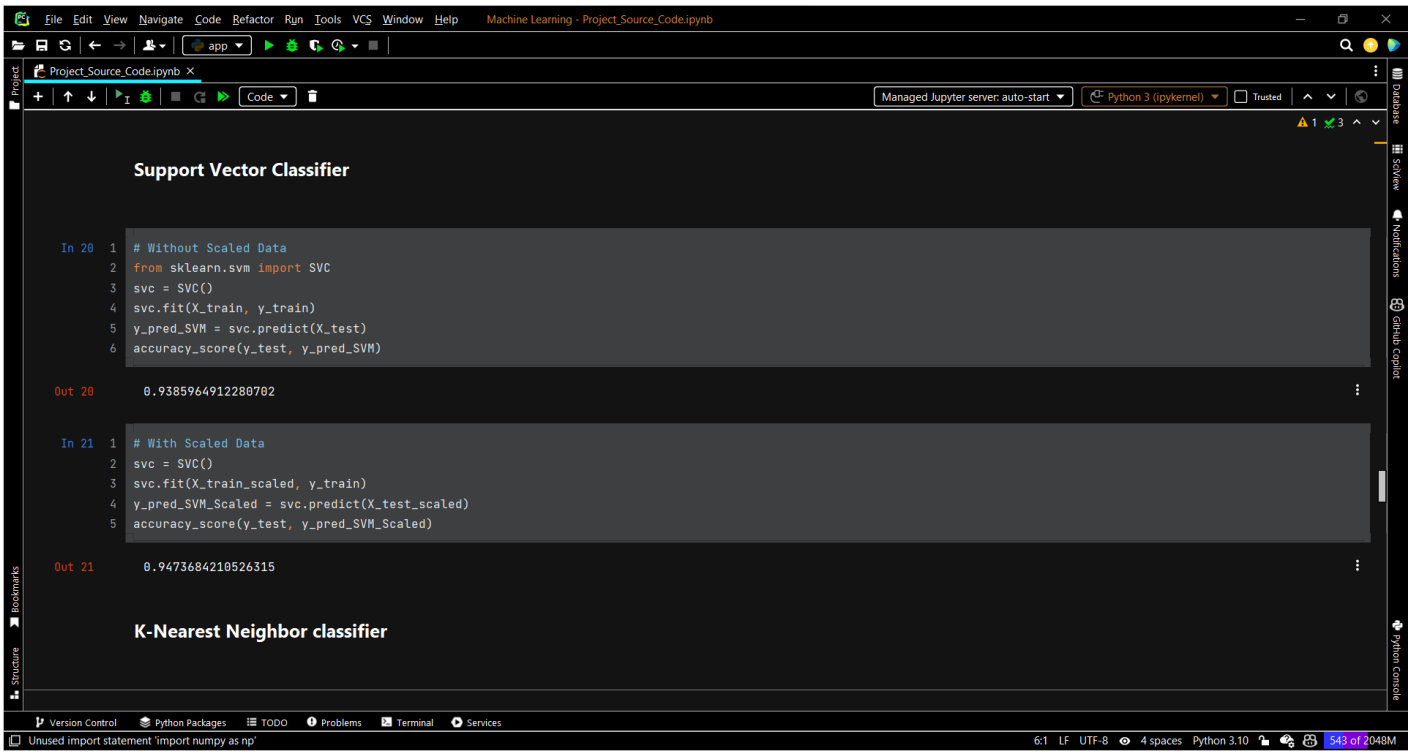
Model Selection

```
In 17  1  from sklearn.metrics import accuracy_score
```

Logistic Regression

```
In 18  1  # Without Scaled Data
       2  from sklearn.linear_model import LogisticRegression
       3  logreg = LogisticRegression()
       4  logreg.fit(X_train, y_train)
       5  y_pred_LR = logreg.predict(X_test)
       6  accuracy_score(y_test, y_pred_LR)
```

```
C:\Users\Rituraj\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```



Support Vector Classifier

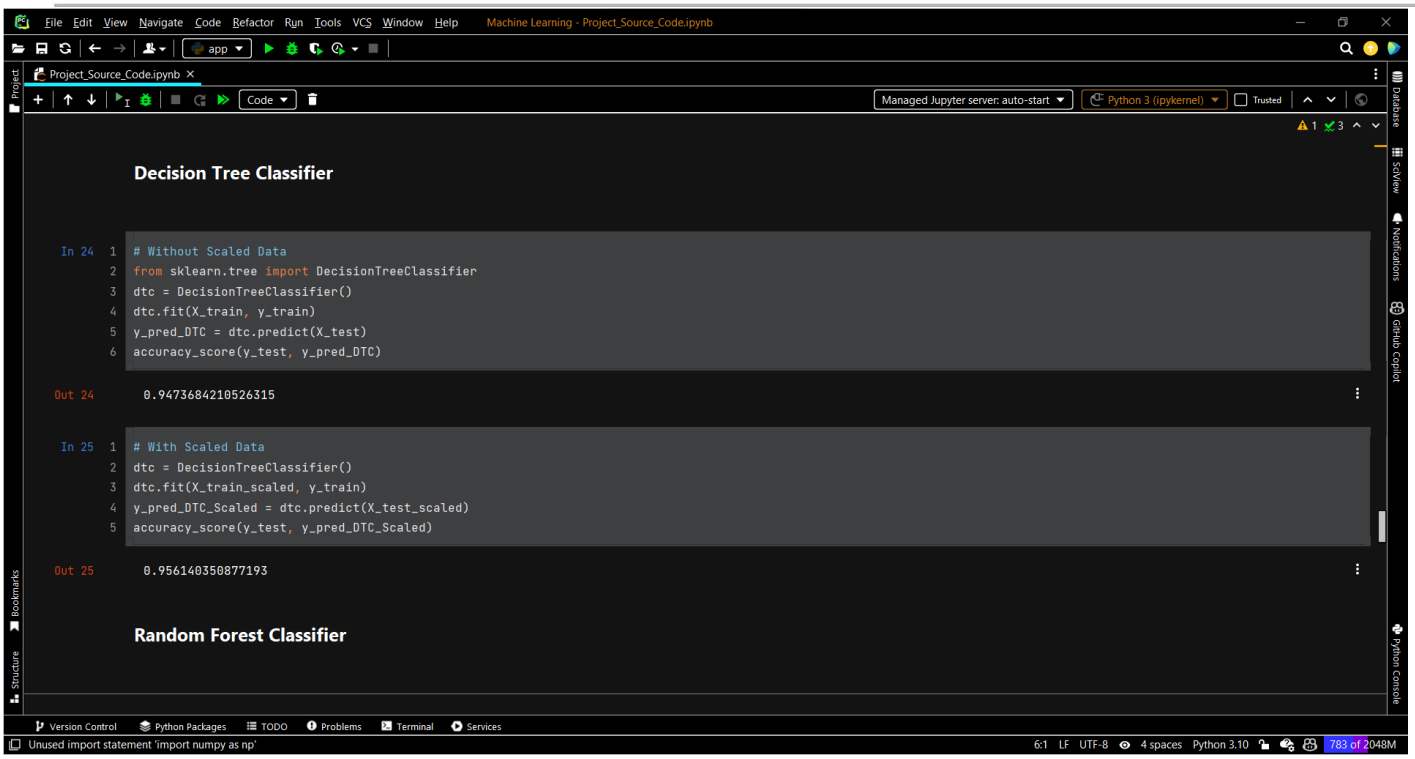
```
In 20  1  # Without Scaled Data
       2  from sklearn.svm import SVC
       3  svc = SVC()
       4  svc.fit(X_train, y_train)
       5  y_pred_SVM = svc.predict(X_test)
       6  accuracy_score(y_test, y_pred_SVM)
```

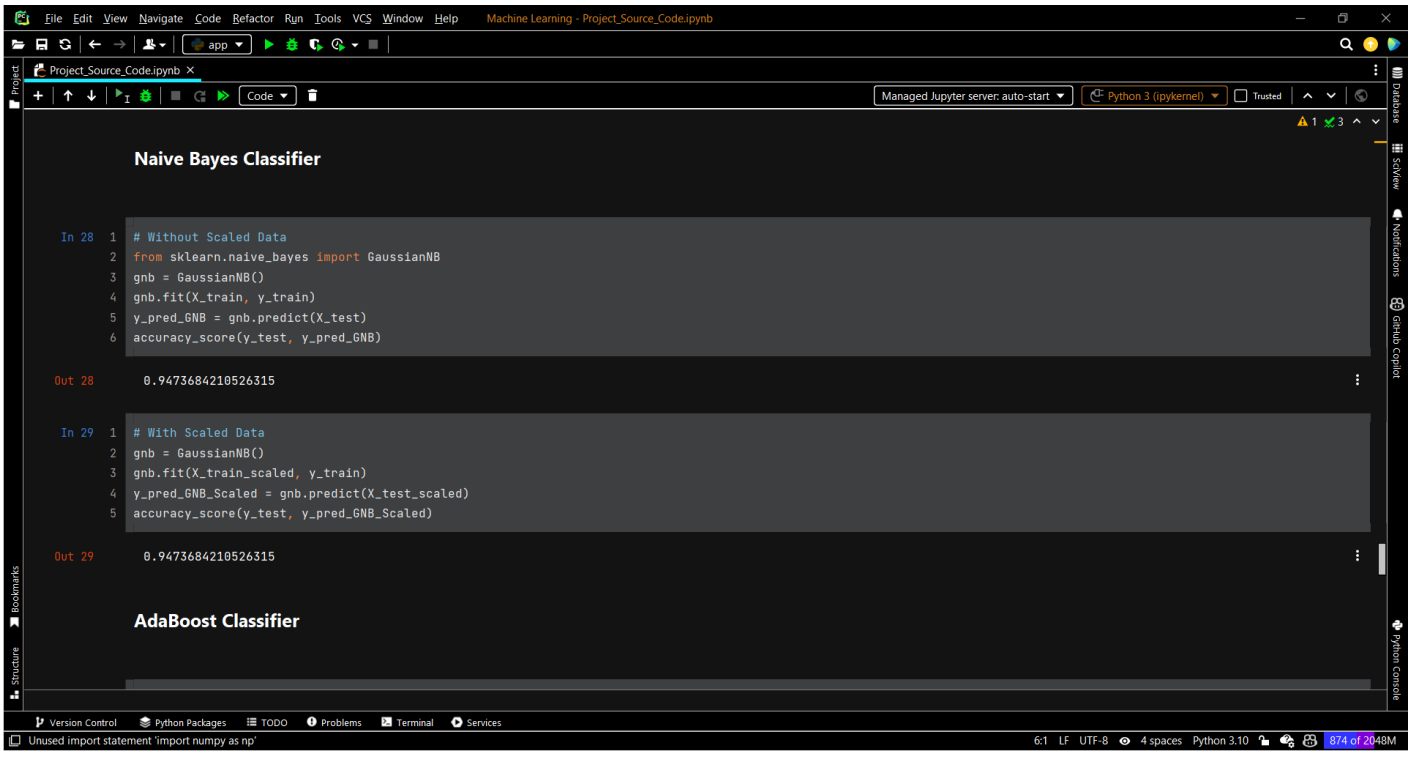
```
Out 20   0.9385964912280702
```

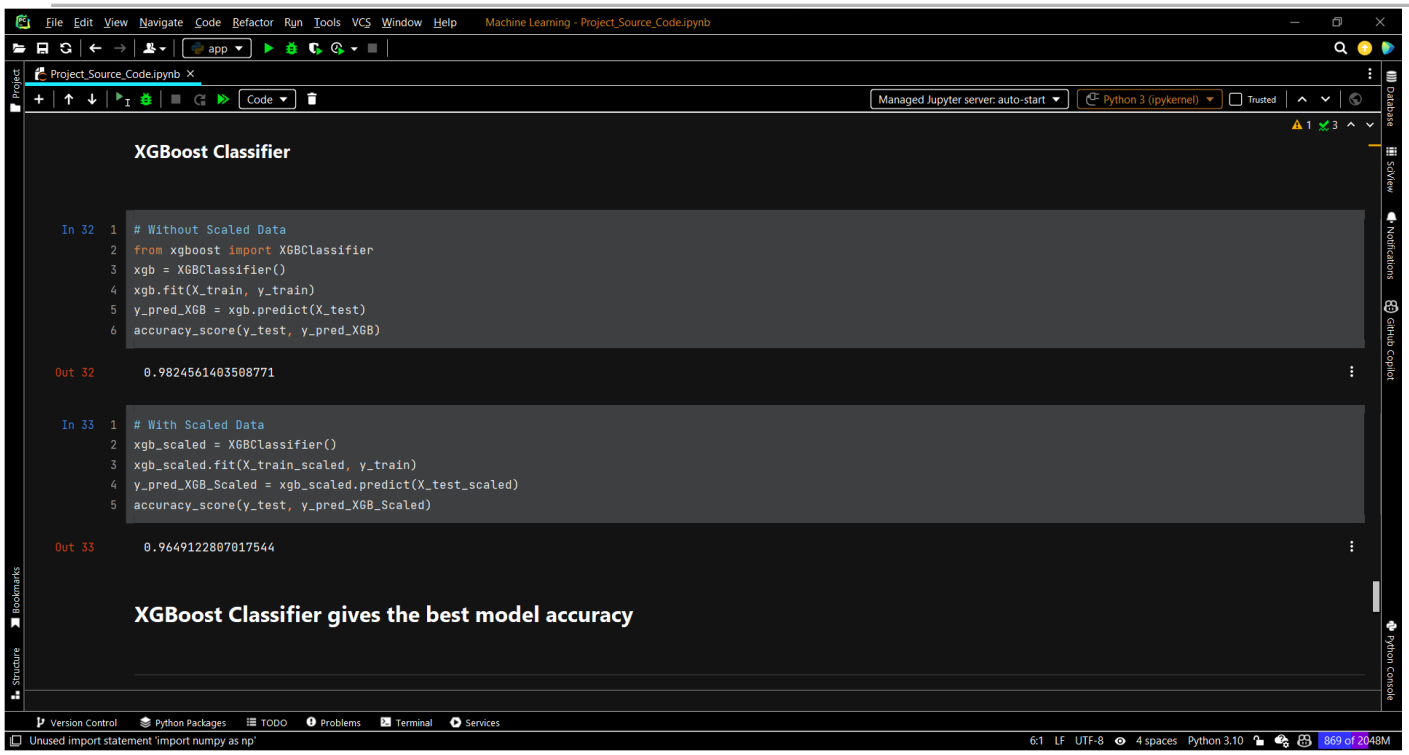
```
In 21  1  # With Scaled Data
       2  svc = SVC()
       3  svc.fit(X_train_scaled, y_train)
       4  y_pred_SVM_Scaled = svc.predict(X_test_scaled)
       5  accuracy_score(y_test, y_pred_SVM_Scaled)
```

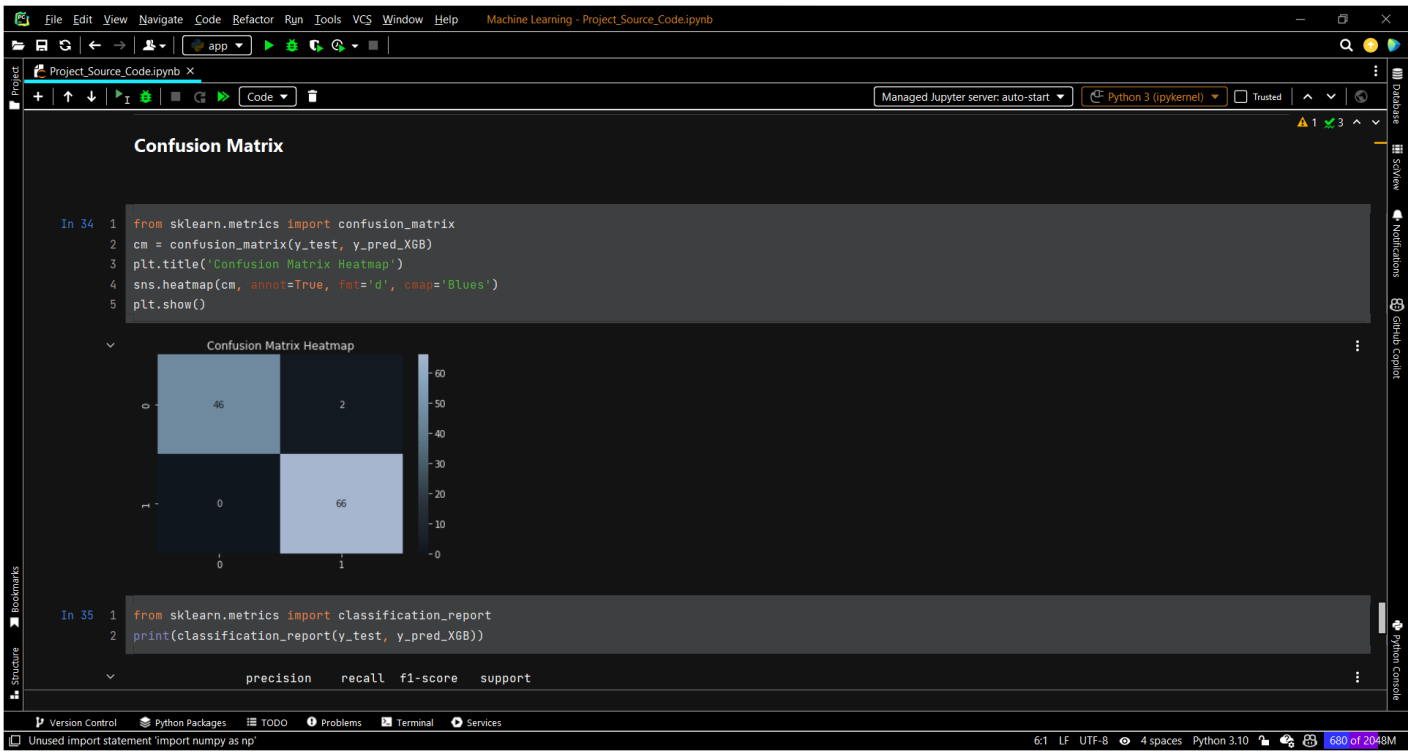
```
Out 21   0.9473684210526315
```

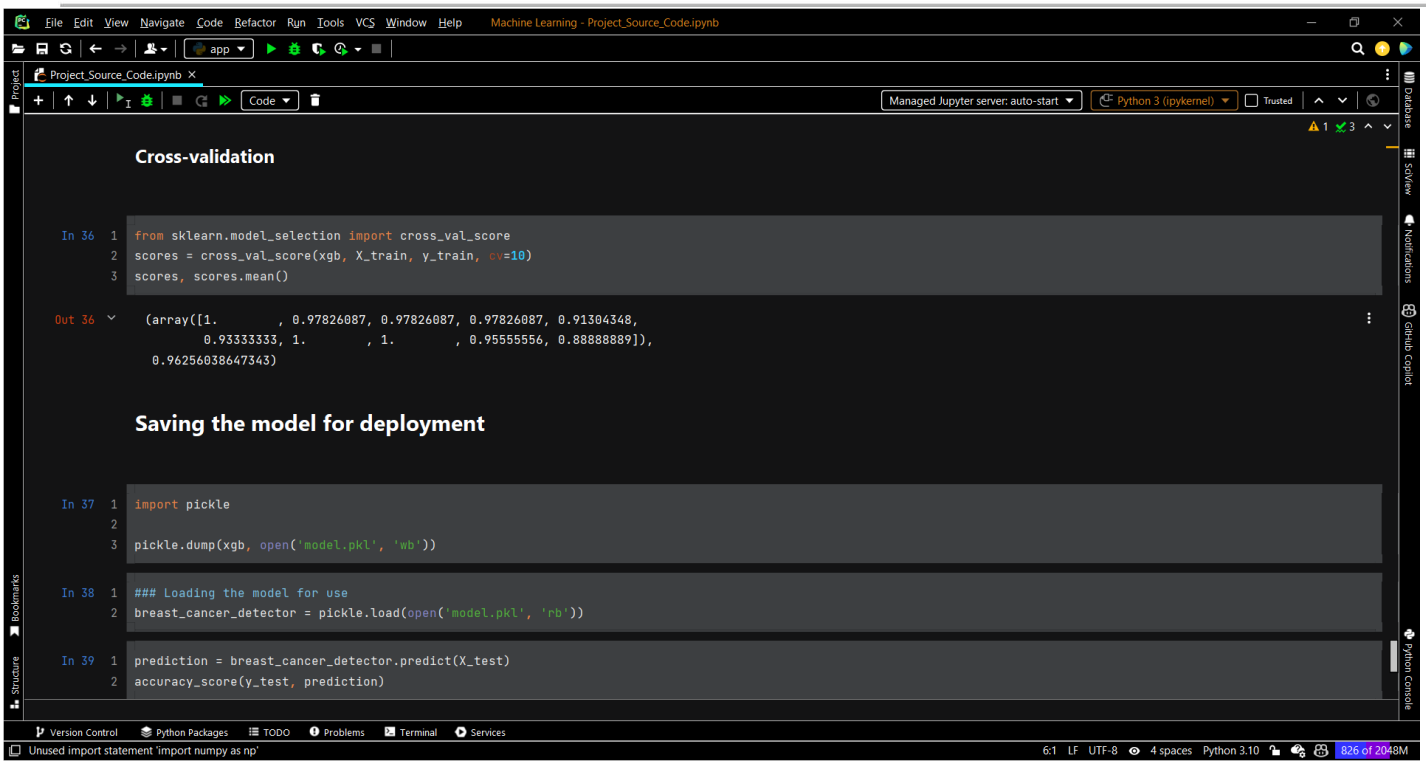
K-Nearest Neighbor classifier


Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Decision Tree Classifier
In 24
# Without Scaled Data
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred_DTC = dtc.predict(X_test)
accuracy_score(y_test, y_pred_DTC)
Out 24
0.9473684210526315
In 25
# With Scaled Data
dtc = DecisionTreeClassifier()
dtc.fit(X_train_scaled, y_train)
y_pred_DTC_Scaled = dtc.predict(X_test_scaled)
accuracy_score(y_test, y_pred_DTC_Scaled)
Out 25
0.956140350877193
Random Forest Classifier
Unused import statement 'import numpy as np'


Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Naive Bayes Classifier
In 28
# Without Scaled Data
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_GNB = gnb.predict(X_test)
accuracy_score(y_test, y_pred_GNB)
Out 28
0.9473684210526315
In 29
# With Scaled Data
gnb = GaussianNB()
gnb.fit(X_train_scaled, y_train)
y_pred_GNB_Scaled = gnb.predict(X_test_scaled)
accuracy_score(y_test, y_pred_GNB_Scaled)
Out 29
0.9473684210526315
AdaBoost Classifier
Unused import statement 'import numpy as np'


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Code
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
XGBoost Classifier
In 32
# Without Scaled Data
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
y_pred_XGB = xgb.predict(X_test)
accuracy_score(y_test, y_pred_XGB)
Out 32
0.9824561403508771
In 33
# With Scaled Data
xgb_scaled = XGBClassifier()
xgb_scaled.fit(X_train_scaled, y_train)
y_pred_XGB_Scaled = xgb_scaled.predict(X_test_scaled)
accuracy_score(y_test, y_pred_XGB_Scaled)
Out 33
0.9649122807017544
XGBoost Classifier gives the best model accuracy
Version Control
Python Packages
TODO
Problems
Terminal
Services
Unused import statement 'import numpy as np'
6:1 LF UTF-8 4 spaces Python 3.10
869 of 2048M


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Machine Learning - Project_Source_Code.ipynb
Project_Source_Code.ipynb
Code
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
Confusion Matrix
In 34
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_XGB)
plt.title('Confusion Matrix Heatmap')
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.show()
Confusion Matrix Heatmap
46
2
0
66
In 35
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_XGB))
precision recall f1-score support
Version Control
Python Packages
TODO
Problems
Terminal
Services
Unused import statement 'import numpy as np'
6:1 LF UTF-8 4 spaces Python 3.10
680 of 2048M



2. Training House Price Prediction Model

Train the model to predict the sales price for each house.

Data fields

Here's a brief version of the data variables:

- SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to the property
- LotArea: Lot size in square feet

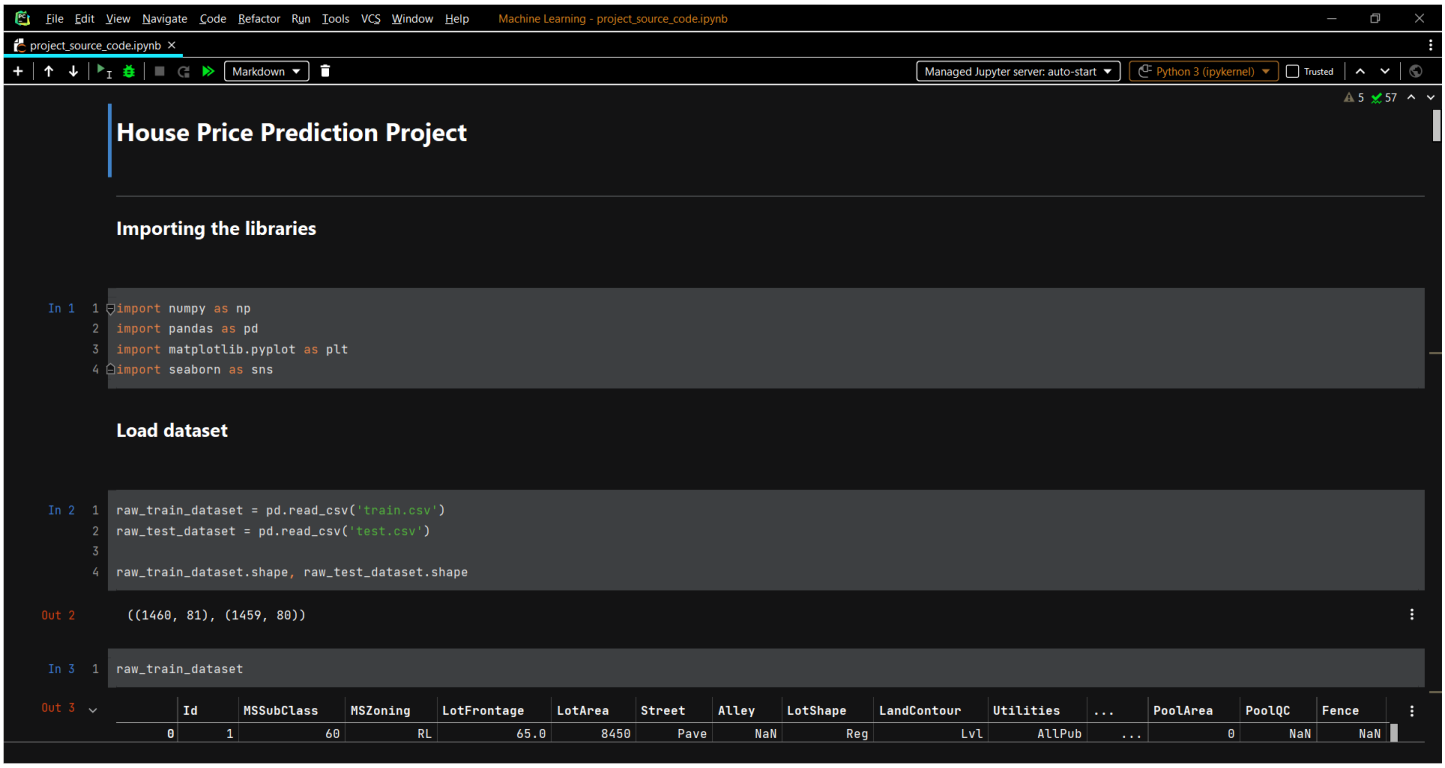
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of the property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to main road or railroad
- Condition2: Proximity to the main road or railroad (if a second is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Overall material and finish quality
- OverallCond: Overall condition rating
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on the house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet

- ExterQual: Exterior material quality
- ExterCond: Present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Height of the basement
- BsmtCond: General condition of the basement
- BsmtExposure: Walkout or garden level basement walls
- BsmtFinType1: Quality of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Quality of second finished area (if present
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of the basement area
- TotalBsmtSF: Total square feet of basement area\
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second-floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms

- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Number of bedrooms above basement level
- Kitchen: Number of kitchens
- KitchenQual: Kitchen Quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality rating
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three-season porch area in square feet
- ScreenPorch: Screen porch area in square feet

- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence Quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold
- YrSold: Year Sold
- SaleType: Type of sale
- SaleCondition: Condition of sale

Model Training Github File: https://github.com/RITURAJRAMAN/House-Price-Prediction



File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Data Integration

In 5
```
integrated_dataset = pd.concat([raw_train_dataset, raw_test_dataset])

org_integrated_dataset = integrated_dataset.copy()
integrated_dataset.shape
```

Out 5
```
(2919, 81)
```

Setting 'Id' variable as index

In 6
```
integrated_dataset.set_index('Id', inplace=True)
org_integrated_dataset.set_index('Id', inplace=True)
```

In 7
```
integrated_dataset.tail()
```

Out 7

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	PoolArea	PoolQC	Fence
2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	I
2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	I
2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	I
2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	Mn
2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	Inside	...	0	NaN	I

5 rows × 80 columns Open in new tab

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Brief information of data

In 8
```
integrated_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919 entries, 1 to 2919
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MSSubClass     2919 non-null   int64
 1   MSZoning       2915 non-null   object
 2   LotFrontage    2433 non-null   float64
 3   LotArea        2919 non-null   int64
 4   Street         2919 non-null   object
 5   Alley          198 non-null    object
```

In 9
```
int_variables = integrated_dataset.select_dtypes(include=['int64']).columns
int_variables.tolist()
```

Out 9
```
['MSSubClass',
 'LotArea',
 'OverallQual',
 'OverallCond',
 'YearBuilt',
 'YearRemodAdd',
 '1stFlrSF',
 '2ndFlrSF',
 'LowQualFinSF',
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Statistical Information of data

In 12
```
integrated_dataset.describe()
```

Out 12

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodD
count	2919.000000	2433.000000	2919.000000	2919.000000	2919.000000	2919.000000	2919.000000	2896.000000	2918.000000	2918.000000	...	291
mean	57.137718	69.305795	10168.114080	6.089072	5.564577	1971.312778	1984.264474	102.201312	441.423235	49.582248	...	9
std	42.517628	23.344905	7886.996359	1.409947	1.113131	30.291442	20.894344	179.334253	455.610826	169.205611	...	12
min	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	0.000000	...	
25%	20.000000	59.000000	7478.000000	5.000000	5.000000	1953.500000	1965.000000	0.000000	0.000000	0.000000	...	
50%	50.000000	68.000000	9453.000000	6.000000	5.000000	1973.000000	1993.000000	0.000000	368.500000	0.000000	...	
75%	70.000000	80.000000	11570.000000	7.000000	6.000000	2001.000000	2004.000000	164.000000	733.000000	0.000000	...	16
max	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1526.000000	...	142

8 rows × 37 columns Open in new tab

Handling missing values

Visualise Null/Missing values

In 13
```
plt.figure(figsize=(25, 16))
sns.heatmap(integrated_dataset.isnull())
plt.savefig('heatmap_missing.png')
plt.show()
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Get the null percentage of each features

In 14
```
null_percent = integrated_dataset.isnull().sum() / integrated_dataset.shape[0] * 100
null_percent
```

Out 14
```
MSZoning          0.137033
LotFrontage      16.649538
LotArea           0.000000
Street            0.000000
                   ...
MoSold            0.000000
YrSold            0.000000
SaleType          0.034258
SaleCondition     0.000000
SalePrice        49.982871
Length: 80, dtype: float64
```

Checking for variables having null values more than 50%

In 15
```
null_percent[null_percent > 50]
```

Out 15
```
Alley          93.216855
PoolQC         99.657417
Fence          80.438506
MiscFeature    96.402878
dtype: float64
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

```
SaleCondition      0.000000
Length: 74, dtype: float64
```

According to data_description, values that seem to be missing and represented as 'NA' in the dataset are actually not missing values, so we will not drop these variables and fill those with a constant value

Imputing missing values

In 18

```
missing_value_variables = null_percent[null_percent > 0]
print(len(missing_value_variables))
missing_value_variables
```

```
35
```

Out 18

```
MSZoning         0.137033
LotFrontage     16.649538
Alley           93.216855
Utilities        0.068517
Exterior1st      0.034258
Exterior2nd      0.034258
MasVnrType       0.822199
MasVnrArea       0.787941
BsmtQual         2.774923
BsmtCond         2.809181
BsmtExposure     2.809181
```

Handling Integer variables

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Handling Integer variables

In 19

```
int_missing_variables = missing_value_variables[missing_value_variables.keys().isin(int_variables)]
int_missing_variables
```

Out 19

```
Series([], dtype: float64)
```

No Missing values in Integer variables

Handling Float variables

In 20

```
float_missing_variables = missing_value_variables[missing_value_variables.keys().isin(float_variables)]
float_missing_variables
```

Out 20

```
LotFrontage     16.649538
MasVnrArea       0.787941
BsmtFinSF1       0.034258
BsmtFinSF2       0.034258
BsmtUnfSF        0.034258
TotalBsmtSF      0.034258
BsmtFullBath     0.068517
BsmtHalfBath     0.068517
GarageYrBlt      5.447071
GarageCars       0.034258
GarageArea       0.034258
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

MasVnrArea MasVnrArea

5 57

In 27
```
masvnrarea_const = 0
integrated_dataset.MasVnrArea.replace(np.nan, masvnrarea_const, inplace=True)
integrated_dataset.MasVnrArea.isnull().sum()
```

Out 27
```
0
```

Handling BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath, BsmtHalfBath, GarageYrBlt, GarageCars, GarageArea

In 28
```
num_var_const = ['BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath', 'GarageYrBlt',
                 'GarageCars', 'GarageArea']
for var in num_var_const:
    integrated_dataset[var].replace(np.nan, 0, inplace=True)
```

Handling categorical variables

In 29
```
cat_missing_variables = missing_value_variables[missing_value_variables.keys().isin(cat_variables)]
cat_missing_variables
```

Out 29
```
MSZoning        0.137033
Alley          93.216855
Utilities       0.068517
Exterior1st     0.034258
Exterior2nd     0.034258
MasVnrType      0.822199
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

5 57

Handling with 'Alley', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond','FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'

In 31
```
cat_var_const = ['Alley', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'GarageType',
                 'GarageFinish', 'GarageQual', 'GarageCond', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature']
for var in cat_var_const:
    integrated_dataset[var].replace(np.nan, 'NA', inplace=True)
```

Cleaned dataset

In 32
```
plt.figure(figsize=(25, 16))
sns.heatmap(integrated_dataset.isnull())
plt.savefig('heatmap_cleaned.png')
plt.show()
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Feature Transformation

Converting numerical feature to categorical feature

In 33
```
num_to_cat = ['MSSubClass', 'YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']
for var in num_to_cat:
    integrated_dataset[var] = integrated_dataset[var].astype(str)
for var in num_to_cat:
    print(var, integrated_dataset[var].dtypes)
```

```
MSSubClass object
YearBuilt object
YearRemodAdd object
GarageYrBlt object
YrSold object
```

In 34
```
import calendar

integrated_dataset['MoSold'] = integrated_dataset['MoSold'].apply(lambda x: calendar.month_abbr[x])
integrated_dataset.MoSold.unique()
```

Out 34
```
array(['Feb', 'May', 'Sep', 'Dec', 'Oct', 'Aug', 'Nov', 'Apr', 'Jan',
       'Jul', 'Mar', 'Jun'], dtype=object)
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Converting categorical feature to numerical feature

Ordinal Encoding

In 35
```
from pandas.api.types import CategoricalDtype
```

In 36
```
integrated_dataset['BsmtCond'].unique()
```

Out 36
```
array(['TA', 'Gd', 'NA', 'Fa', 'Po'], dtype=object)
```

In 37
```
integrated_dataset['BsmtCond'] = integrated_dataset['BsmtCond'].astype(
    CategoricalDtype(categories=['NA', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
```

In 38
```
integrated_dataset['BsmtCond'].unique()
```

Out 38
```
array([3, 4, 0, 2, 1], dtype=int8)
```

In 39
```
integrated_dataset['BsmtExposure'] = integrated_dataset['BsmtExposure'].astype(
    CategoricalDtype(categories=['NA', 'Mn', 'Av', 'Gd'], ordered=True)).cat.codes
integrated_dataset['BsmtFinType1'] = integrated_dataset['BsmtFinType1'].astype(
    CategoricalDtype(categories=['NA', 'Unf', 'LwQ', 'Rec', 'BLQ', 'ALQ', 'GLQ'], ordered=True)).cat.codes
integrated_dataset['BsmtFinType2'] = integrated_dataset['BsmtFinType2'].astype(
    CategoricalDtype(categories=['NA', 'Unf', 'LwQ', 'Rec', 'BLQ', 'ALQ', 'GLQ'], ordered=True)).cat.codes
integrated_dataset['BsmtQual'] = integrated_dataset['BsmtQual'].astype(
    CategoricalDtype(categories=['NA', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['ExterQual'] = integrated_dataset['ExterQual'].astype(
    CategoricalDtype(categories=['Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['ExterCond'] = integrated_dataset['ExterCond'].astype(
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

In 39

```
integrated_dataset['BsmtExposure'] = integrated_dataset['BsmtExposure'].astype(
    CategoricalDtype(categories=['NA', 'Mn', 'Av', 'Gd'], ordered=True)).cat.codes
integrated_dataset['BsmtFinType1'] = integrated_dataset['BsmtFinType1'].astype(
    CategoricalDtype(categories=['NA', 'Unf', 'LwQ', 'Rec', 'BLQ', 'ALQ', 'GLQ'], ordered=True)).cat.codes
integrated_dataset['BsmtFinType2'] = integrated_dataset['BsmtFinType2'].astype(
    CategoricalDtype(categories=['NA', 'Unf', 'LwQ', 'Rec', 'BLQ', 'ALQ', 'GLQ'], ordered=True)).cat.codes
integrated_dataset['BsmtQual'] = integrated_dataset['BsmtQual'].astype(
    CategoricalDtype(categories=['NA', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['ExterQual'] = integrated_dataset['ExterQual'].astype(
    CategoricalDtype(categories=['Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['ExterCond'] = integrated_dataset['ExterCond'].astype(
    CategoricalDtype(categories=['Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['Functional'] = integrated_dataset['Functional'].astype(
    CategoricalDtype(categories=['Sal', 'Sev', 'Maj2', 'Maj1', 'Mod', 'Min2', 'Min1', 'Typ'], ordered=True)).cat.codes
integrated_dataset['GarageCond'] = integrated_dataset['GarageCond'].astype(
    CategoricalDtype(categories=['NA', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['GarageQual'] = integrated_dataset['GarageQual'].astype(
    CategoricalDtype(categories=['NA', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['GarageFinish'] = integrated_dataset['GarageFinish'].astype(
    CategoricalDtype(categories=['NA', 'Unf', 'RFn', 'Fin'], ordered=True)).cat.codes
integrated_dataset['HeatingQC'] = integrated_dataset['HeatingQC'].astype(
    CategoricalDtype(categories=['Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['KitchenQual'] = integrated_dataset['KitchenQual'].astype(
    CategoricalDtype(categories=['Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['PavedDrive'] = integrated_dataset['PavedDrive'].astype(
    CategoricalDtype(categories=['N', 'P', 'Y'], ordered=True)).cat.codes
integrated_dataset['Utilities'] = integrated_dataset['Utilities'].astype(
    CategoricalDtype(categories=['ELO', 'NASeWa', 'NASeWr', 'AllPub'], ordered=True)).cat.codes
integrated_dataset['PoolQC'] = integrated_dataset['PoolQC'].astype(
    CategoricalDtype(categories=['NA', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
integrated_dataset['FireplaceQu'] = integrated_dataset['FireplaceQu'].astype(
    CategoricalDtype(categories=['NA', 'Po', 'Fa', 'TA', 'Gd', 'Ex'], ordered=True)).cat.codes
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

One-Hot Encoding

In 40

```
integrated_dataset_encode = integrated_dataset.copy()

object_variables = integrated_dataset_encode.select_dtypes(include='object').columns.tolist()
object_variables
```

Out 40

```
['MSSubClass',
 'MSZoning',
 'Street',
 'Alley',
 'LotShape',
 'LandContour',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
```

In 41

```
print("Shape before encoding: ", integrated_dataset_encode.shape)
```

```
Shape before encoding:  (2919, 80)
```

In 42

```
integrated_dataset_encode = pd.get_dummies(integrated_dataset_encode, columns=object_variables,
                                           prefix_sep=object_variables, drop_first=True)
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

Markdown | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Splitting data for training

In 45
```
integrated_dataset_encode.shape
```
Out 45
```
(2919, 513)
```

In 46
```
X_train = integrated_dataset_encode[:raw_train_dataset.shape[0]].drop('SalePrice', axis=1)
y_train = integrated_dataset_encode['SalePrice'][:raw_train_dataset.shape[0]]
X_test = integrated_dataset_encode[raw_train_dataset.shape[0]:].drop('SalePrice', axis=1)

X_train.shape, y_train.shape, X_test.shape
```
Out 46
```
((1460, 512), (1460,), (1459, 512))
```

Feature Scaling

In 47
```
from sklearn.preprocessing import StandardScaler

sc_object = StandardScaler()
sc_object.fit(X_train)
X_train = sc_object.transform(X_train)
X_test = sc_object.transform(X_test)
```

In 48
```
X_train
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Machine Learning - project_source_code.ipynb

project_source_code.ipynb

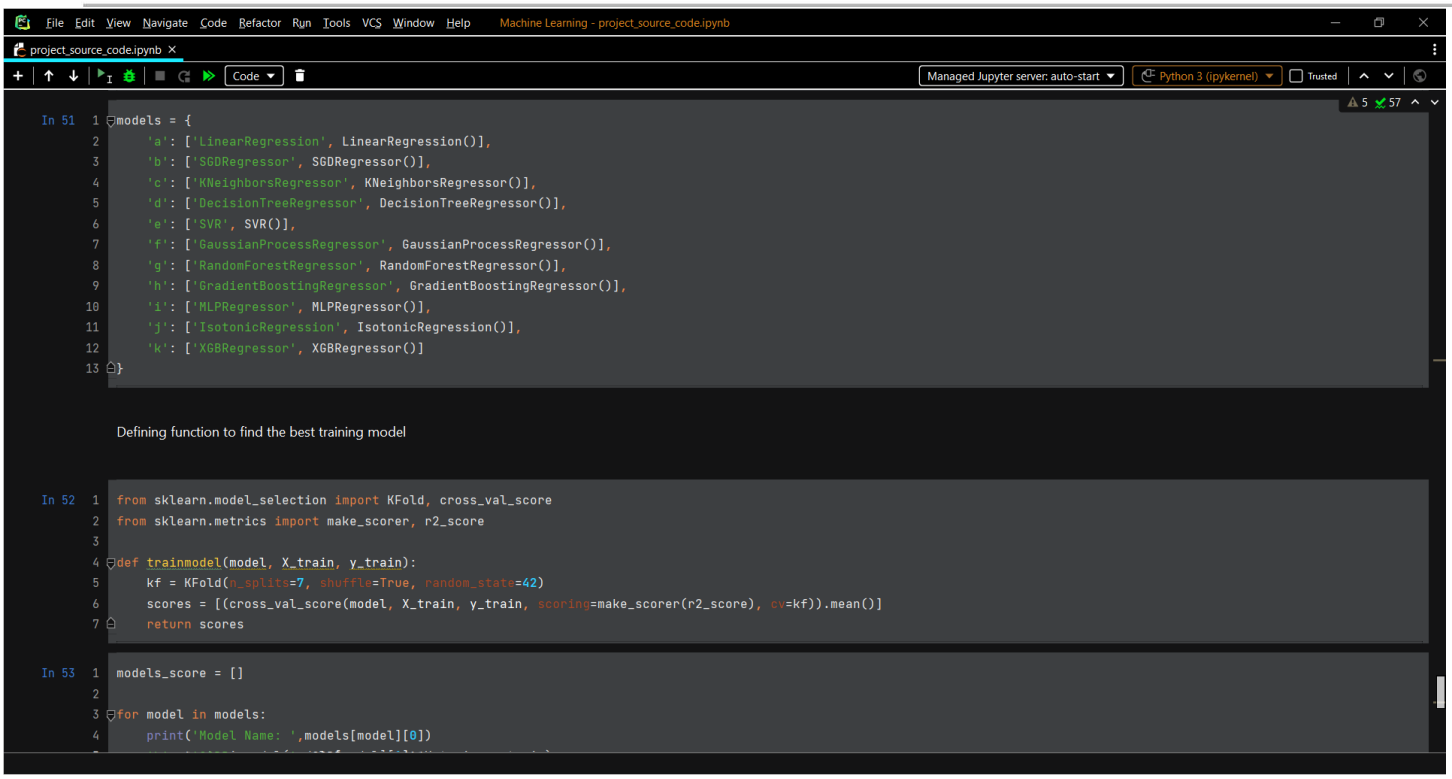
Code | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

Model Training

Importing Model Libraries

In 50
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.svm import SVR

from sklearn.gaussian_process import GaussianProcessRegressor

from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.neural_network import MLPRegressor

from sklearn.isotonic import IsotonicRegression

from xgboost import XGBRegressor
```

In 51
```
models = {
    'a': ['LinearRegression', LinearRegression()],
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Machine Learning - project_source_code.ipynb
project_source_code.ipynb
Code
Managed Jupyter server: auto-start
Python 3 (ipykernel)
Trusted
In 51
models = {
'a': ['LinearRegression', LinearRegression()],
'b': ['SGDRegressor', SGDRegressor()],
'c': ['KNeighborsRegressor', KNeighborsRegressor()],
'd': ['DecisionTreeRegressor', DecisionTreeRegressor()],
'e': ['SVR', SVR()],
'f': ['GaussianProcessRegressor', GaussianProcessRegressor()],
'g': ['RandomForestRegressor', RandomForestRegressor()],
'h': ['GradientBoostingRegressor', GradientBoostingRegressor()],
'i': ['MLPRegressor', MLPRegressor()],
'j': ['IsotonicRegression', IsotonicRegression()],
'k': ['XGBRegressor', XGBRegressor()]
}
Defining function to find the best training model
In 52
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import make_scorer, r2_score
def trainmodel(model, X_train, y_train):
kf = KFold(n_splits=7, shuffle=True, random_state=42)
scores = [(cross_val_score(model, X_train, y_train, scoring=make_scorer(r2_score), cv=kf)).mean()]
return scores
In 53
models_score = []
for model in models:
print('Model Name: ',models[model][0])

METHODOLOGY ADOPTED

1. Collecting Data:

As we know, machines initially learn from the data that we feed them. It is of the utmost importance to collect reliable data so that your machine learning model can find the correct patterns. The quality of the data that we feed to the machine will determine how accurate your model is. If we have incorrect or outdated data, we will have wrong outcomes or predictions which are not relevant.

We need to make sure that we use data from a reliable source, as it will directly affect the outcome of the model. Good data is relevant, contains very few missing and repeated values, and has a good representation of the various subcategories/classes present.

2. Preparing the Data:

After we have the data, we have to prepare it. It can be done by:

- Putting together all the data we have and randomizing it. This helps make sure that data is evenly distributed, and that the ordering does not affect the learning process.
- Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. we might even have to restructure the dataset and change the rows and columns or index of rows and columns.
- Visualize the data to understand how it is structured and understand the relationship between various variables and classes present.
- Splitting the cleaned data into two sets - a training set and a testing set. The training set is the set your model learns from. A testing set is used to check the accuracy of your model after training.

3. Choosing a Model:

A machine learning model determines the output we get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Over the years, scientists and engineers developed various models suited for different tasks like speech recognition, image recognition, prediction, etc. Apart from this, we also have to see if our model is suited for numerical or categorical data and choose accordingly.

4. Training the Model:

Training is the most important step in machine learning. In training, we pass the prepared data to our machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.

5. Evaluating the Model:

After training our model, we have to check how it performs. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that we split our data into earlier. If testing was done on the same data which is used for training, we will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.

When used on testing data, we accurately measure how your model will perform and its speed.

6. Parameter Tuning:

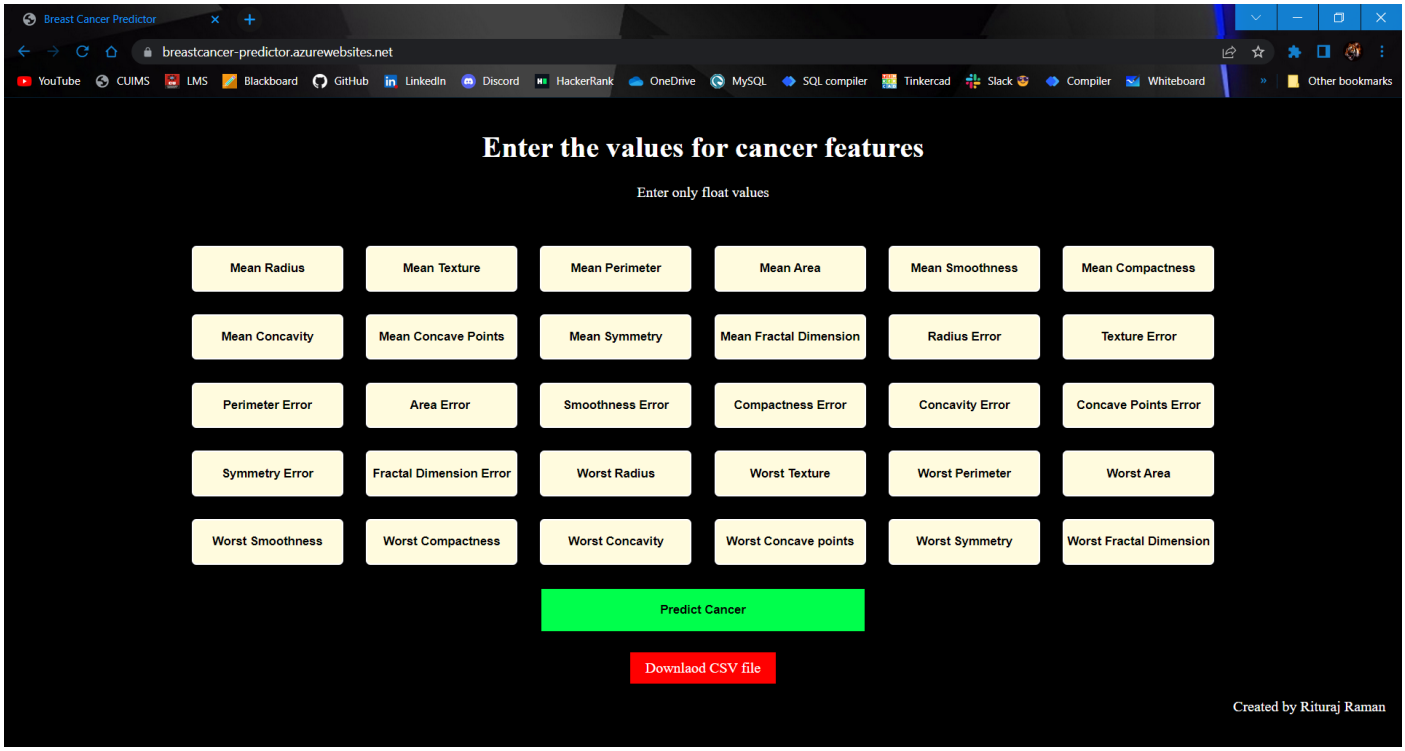
Once we have created and evaluated our model, see if its accuracy can be improved in any way. This is done by tuning the parameters present in the model. Parameters are the variables in the model that the programmer generally decides. The accuracy will be the maximum at a particular parameter value. Parameter tuning refers to finding these values.

7. Making Predictions

In the end, we can use our model on unseen data to make predictions accurately.

RESULTS AND DISCUSSIONS

1. Breast cancer prediction web app



CONCLUSIONS AND SCOPE FOR FUTURE STUDY

Machine Learning is a technique of training machines to perform the activities a human brain can do i.e. machines can be trained to perform human activities in several areas and can aid humans in living better lives.

We can perform different machine learning algorithms to train the machines to perform the different actions/predictions.

There can be two types of machine learning algorithms, Supervised or Unsupervised Machine Learning. There are a lot of ML algorithms to train the ML models.