Contents

[Hand Gesture Recognition 1](#_Toc130398566)

[Model Builder for .NET 8](#_Toc130398567)

[Installation 8](#_Toc130398568)

[Create your app 8](#_Toc130398569)

[Add machine learning 9](#_Toc130398570)

[Pick a scenario 10](#_Toc130398571)

[Download and add data 12](#_Toc130398572)

[Add data 12](#_Toc130398573)

[Train your model 13](#_Toc130398574)

[Training results 13](#_Toc130398575)

[Evaluate your model 14](#_Toc130398576)

[Try out your model 14](#_Toc130398577)

[Generate code 15](#_Toc130398578)

[Consume your model 16](#_Toc130398579)

## Hand Gesture Recognition

This section discusses a technique for using predictive models to detect different types of hand movement. The main reference used for this tutorial is provided by **MediaPipe** who are the makers of the hand gesture framework called “**Mediapipe Hands**”.

<https://google.github.io/mediapipe/solutions/hands.html>

Documentation for other types of movement detection is located at their Git repository.

MediaPipe offers various solutions for detecting the shape and motion of objects. For this project, MediaPipe Hands will be used to track hand and finger motions.

* *Hand localization with the palm detection*

We are not going to go deep in this palm detection but what this basically does is that unlike faces that have high contrast patterns, hands are difficult to detect solely relying on their visual features. Instead, they provided additional context like arm, body, or person features, to perform the hand localization

* *Hand knuckles detection and landmarks assignment*

After the palm detection over the whole image, the hand landmark model performs localization of 21 3D hand-knuckle coordinates. Which is called hand landmarks (Fig 2.)

Text

Description automatically generated

*From mediaPipe website https://google.github.io/mediapipe/*

1. Recognize the hand using the MediaPipe library
2. Connect the dots and calculate the angel of each knuckle (landmarks) to determine which join has been how much banded
3. With the calculated angle, recognize the gesture
4. Decide the winner

A picture containing logo

Description automatically generated

Example 1: Building a Rock-Scissors-Paper game

This example is based heavily on work submitted by a student last year. She did such an amazing job that almost all of the original presentation remains in tact.

Import all the required libraries

|  |
| --- |
| import cv2 import mediapipe as mp import numpy as np |

Set the number of maximum hand to be recognized and extract required the gestures for the project.

|  |
| --- |
| max\_num\_hands = 1 # number of maximum hand to be recognized  rps\_gesture = {0:'rock', 5:'paper', 9:'scissors'} # map the gestures for rock-paper-scissors |

Use the MediaPipe hands model

|  |
| --- |
| mp\_hands = mp.solutions.hands mp\_drawing = mp.solutions.drawing\_utils hands = mp\_hands.Hands( # Initialize the hand detection module  max\_num\_hands=max\_num\_hands,  min\_detection\_confidence=0.5,  min\_tracking\_confidence=0.5) # default/optimum values of min and max detection confidence level are 0.5 |

Read the pre-set data and train a KNN model. The csv data is at the end of this tutorial.

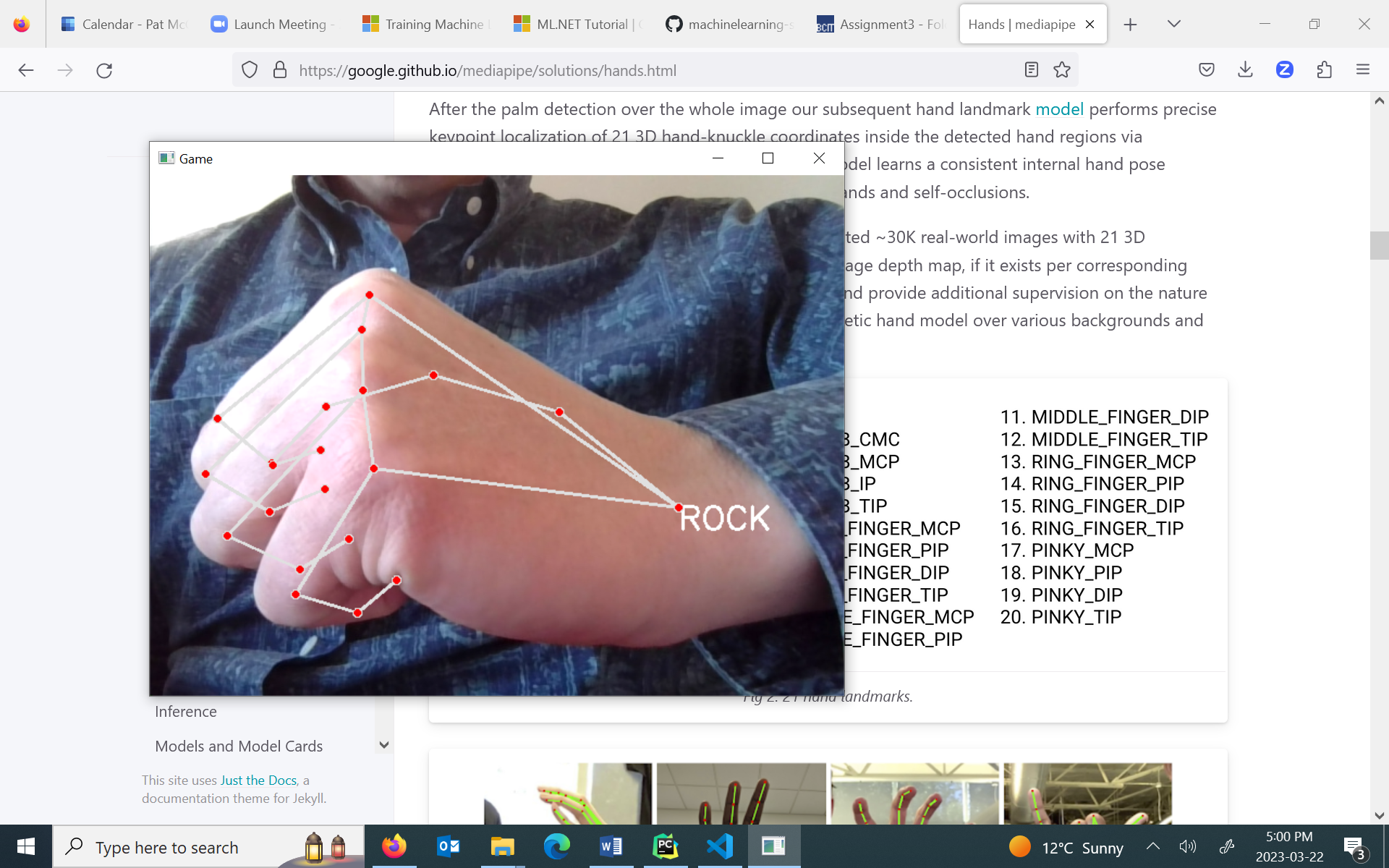
|  |
| --- |
| # Gesture recognition model file = np.genfromtxt('data/gesture\_train.csv', delimiter=',') angle = file[:, :-1].astype(np.float32) label = file[:, -1].astype(np.float32) knn = cv2.ml.KNearest\_create() knn.train(angle, cv2.ml.ROW\_SAMPLE, label) |

Open the web camera

|  |
| --- |
| cap = cv2.VideoCapture(0) |

Recognize the hand and the gesture with landmarks. Then perform the game logic and show the result.

|  |
| --- |
| while cap.isOpened():  ret, img = cap.read()  if not ret:  continue   img = cv2.flip(img, 1)  img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)   result = hands.process(img)   img = cv2.cvtColor(img, cv2.COLOR\_RGB2BGR)   if result.multi\_hand\_landmarks is not None:  rps\_result = [] # save the results of the rock-scissors-paper game and hands coordination in the array   for res in result.multi\_hand\_landmarks:  joint = np.zeros((21, 3))  for j, lm in enumerate(res.landmark):  joint[j] = [lm.x, lm.y, lm.z]   # Compute angles between joints  v1 = joint[[0, 1, 2, 3, 0, 5, 6, 7, 0, 9, 10, 11, 0, 13, 14, 15, 0, 17, 18, 19], :] # Parent joint  v2 = joint[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20], :] # Child joint  v = v2 - v1 # [20,3]  # Normalize v  v = v / np.linalg.norm(v, axis=1)[:, np.newaxis]   # Get angle using arcos of dot product  angle = np.arccos(np.einsum('nt,nt->n',  v[[0, 1, 2, 4, 5, 6, 8, 9, 10, 12, 13, 14, 16, 17, 18], :],  v[[1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 17, 18, 19], :])) # [15,]   angle = np.degrees(angle) # Convert radian to degree   # Inference gesture  data = np.array([angle], dtype=np.float32)  ret, results, neighbours, dist = knn.findNearest(data, 3)  idx = int(results[0][0])   # Draw gesture result  if idx in rps\_gesture.keys():  org = (int(res.landmark[0].x \* img.shape[1]), int(res.landmark[0].y \* img.shape[0]))  cv2.putText(img, text=rps\_gesture[idx].upper(), org=(org[0], org[1] + 20),  fontFace=cv2.FONT\_HERSHEY\_SIMPLEX, fontScale=1, color=(255, 255, 255), thickness=2)   rps\_result.append({  'rps': rps\_gesture[idx], # save the type of gesture  'org': org # save the locations of the two hands  })   mp\_drawing.draw\_landmarks(img, res, mp\_hands.HAND\_CONNECTIONS)   # Who wins?  if len(rps\_result) >= 2: # if the hands are more than two  winner = None  text = ''   # game logic   if rps\_result[0]['rps'] == 'rock':  if rps\_result[1]['rps'] == 'rock':  text = 'Tie'  elif rps\_result[1]['rps'] == 'paper':  text = 'Paper wins';  winner = 1  elif rps\_result[1]['rps'] == 'scissors':  text = 'Rock wins';  winner = 0  elif rps\_result[0]['rps'] == 'paper':  if rps\_result[1]['rps'] == 'rock':  text = 'Paper wins';  winner = 0  elif rps\_result[1]['rps'] == 'paper':  text = 'Tie'  elif rps\_result[1]['rps'] == 'scissors':  text = 'Scissors wins';  winner = 1  elif rps\_result[0]['rps'] == 'scissors':  if rps\_result[1]['rps'] == 'rock':  text = 'Rock wins';  winner = 1  elif rps\_result[1]['rps'] == 'paper':  text = 'Scissors wins';  winner = 0  elif rps\_result[1]['rps'] == 'scissors':  text = 'Tie'   # show the winner in text  if winner is not None:  cv2.putText(img, text='Winner',  org=(rps\_result[winner]['org'][0], rps\_result[winner]['org'][1] + 70),  fontFace=cv2.FONT\_HERSHEY\_SIMPLEX, fontScale=2, color=(0, 255, 0), thickness=3)  cv2.putText(img, text=text, org=(int(img.shape[1] / 2), 100), fontFace=cv2.FONT\_HERSHEY\_SIMPLEX,  fontScale=2, color=(0, 0, 255), thickness=3)   cv2.imshow('Game', img)  if cv2.waitKey(1) == ord('q'):  break |



Exercise 1 (4 marks)

Build and run this example. Show a picture of you making a rock or paper. Show your screenshot here. (Scissors did not work well for me but maybe you will have more success)

|  |
| --- |
|  |

Gesture\_train.csv – contains the angles of landmarks and the index for each gesture index

|  |
| --- |
| 32.647995,27.334458,18.777239,32.558145,149.876268,29.578624,37.800344,133.458050,45.085468,32.342115,142.727088,37.366495,26.056262,139.415512,40.519754,0.000000 45.166793,26.117019,18.645196,31.567098,138.965945,20.022370,33.176819,132.806630,21.381263,28.201580,138.854737,14.808465,26.329570,128.394781,23.400235,0.000000 20.826588,35.926397,39.943409,20.257126,88.651968,29.781060,17.812546,102.683664,16.751548,11.247996,103.264677,18.000063,15.584646,82.515342,23.045704,0.000000 31.327256,24.482902,27.073693,22.894414,64.537885,47.775269,16.256466,89.803293,40.146627,9.443338,91.327471,35.448822,14.446064,71.702363,35.047523,0.000000 28.164625,31.028619,23.568194,21.468457,141.750809,21.386416,29.323112,129.772949,38.852031,24.264641,138.309162,33.321451,23.231644,129.452243,30.819853,0.000000 39.493933,30.825017,37.798715,48.789407,101.569152,29.543310,62.408973,102.175635,65.566150,59.301602,112.161872,55.427656,60.565796,102.860590,49.489659,0.000000 46.636410,27.871526,25.808261,22.547515,69.819060,42.556670,16.971363,85.544017,36.496952,11.288500,75.542530,39.251253,16.539423,48.637377,29.019384,0.000000 45.042568,19.961871,11.893033,13.603903,32.278803,27.388893,7.747610,17.797417,5.794515,2.940872,16.492116,1.481793,9.127502,8.934992,6.483260,0.000000 47.299112,32.659637,25.455925,33.674011,52.766652,43.792789,29.175726,97.063906,24.384931,27.473441,104.583461,24.572468,32.064351,94.460182,22.230902,0.000000 46.483112,30.871050,36.277547,40.493482,96.787611,39.591184,38.159227,110.260392,30.181509,30.411843,119.519070,30.006762,26.418003,122.606342,33.310417,0.000000 27.792747,27.346865,18.358880,13.459867,12.675573,14.827358,21.851454,94.371765,35.972556,19.283318,104.185629,31.780360,26.073177,81.116557,33.292727,1.000000 35.320273,26.381613,20.191984,14.668020,7.185043,4.248832,25.126893,121.084548,26.620646,24.078772,117.775156,30.698631,30.541078,89.432634,37.903796,1.000000 26.048195,19.402587,26.436642,6.365824,9.542588,3.241575,28.086456,113.535227,26.468682,20.288474,116.216192,24.670197,26.922405,95.790580,20.527646,1.000000 27.221395,15.414500,25.939750,6.623381,7.608257,2.696879,28.052599,108.763997,27.351137,21.551970,108.780979,26.114410,25.929242,91.324413,21.103923,1.000000 32.014172,10.991831,28.415461,5.728636,6.481493,0.889479,23.010316,90.874638,34.321755,12.829574,92.857042,35.219209,11.395936,68.408045,41.629204,1.000000 35.391177,18.592058,33.007410,4.140826,6.493375,1.912261,25.824411,100.550619,29.616766,19.956852,99.451776,31.702370,24.273458,82.648994,25.282382,1.000000 33.667543,32.918064,19.468711,7.488233,2.724066,7.948471,28.506697,49.497494,13.477383,55.206680,70.580958,23.938099,54.585635,72.790658,27.310205,1.000000 34.297057,31.368970,12.617624,12.278204,1.644287,8.980053,30.469056,26.478147,7.106024,52.598374,70.892878,25.786772,42.428934,65.808129,34.571704,1.000000 37.283942,26.503899,38.901662,12.471588,1.771188,4.433059,38.654754,109.571409,23.595078,35.738030,105.880785,31.167126,25.614191,107.024708,29.663542,1.000000 37.309748,20.163841,29.692652,7.735964,3.484735,2.936218,33.402019,102.752624,25.379006,27.139984,100.369156,26.013738,25.387794,95.077939,19.624609,1.000000 27.064267,4.116995,11.001622,16.486113,10.552898,11.181537,14.660537,7.230117,6.805212,5.992467,1.279764,12.949280,2.583446,5.751485,12.190489,2.000000 31.191134,4.290191,11.017077,14.137684,3.406932,3.979965,14.172432,62.540096,5.982379,4.602033,33.828657,3.623022,7.488203,11.515055,14.876723,2.000000 32.675400,6.051246,9.601219,16.191259,6.049887,3.021566,23.574714,34.476614,52.095947,20.944558,28.680245,10.530155,21.732707,9.850545,3.993437,2.000000 36.983640,10.596499,14.782542,8.568285,5.152676,2.903961,20.725178,85.560392,20.407994,15.760408,73.182706,27.857098,16.365531,47.081374,18.958339,2.000000 28.553979,4.386473,14.044008,14.800291,8.822664,4.434532,12.653980,85.107713,5.685620,4.419841,57.803916,8.104906,9.925122,21.634206,28.194048,2.000000 43.905587,4.436173,6.406616,7.984581,3.269949,4.945221,21.120639,23.389945,11.310605,16.343349,24.295424,15.957736,9.628379,15.356038,23.239797,2.000000 47.227182,2.093504,13.185004,10.198872,9.016928,4.762649,30.488841,87.460182,19.246491,31.417656,83.437891,35.377278,26.301992,71.631192,41.428162,2.000000 38.963707,3.312926,14.764106,13.346643,2.701027,3.122325,15.783439,70.890089,24.271086,16.162595,55.105191,27.125569,14.065205,31.765137,30.163877,2.000000 38.235214,9.765067,12.204059,13.054985,3.186906,6.008764,30.496132,105.006685,19.614097,37.621662,101.843967,33.236817,36.027421,96.934316,33.416261,2.000000 44.894136,5.703140,18.672275,16.862815,7.250208,1.171276,27.349864,91.238617,15.085455,31.288754,83.884860,30.505573,32.303599,72.093530,33.306199,2.000000 33.233165,5.251370,12.361472,7.406677,2.403716,1.809087,14.060716,1.544563,4.514234,31.659243,100.935772,25.510835,35.251030,89.933031,19.148129,3.000000 52.796238,11.266403,14.081873,4.255632,1.900488,2.120873,8.555830,4.423145,2.657591,32.922917,102.226827,22.889829,35.501908,85.247861,24.733276,3.000000 45.530282,6.599183,2.705272,11.851240,11.457306,8.534295,10.935138,3.143677,7.689091,40.015122,108.650040,20.131360,44.474448,95.466861,23.438341,3.000000 40.178526,5.883763,11.293230,9.412355,5.010292,3.067079,17.325150,5.933416,5.999398,39.241775,112.753544,23.859172,37.437907,104.919452,24.762976,3.000000 41.220657,6.328683,11.057584,8.445178,5.794255,1.084988,11.758416,8.098485,5.679773,36.361657,116.513918,16.134616,37.406455,101.434675,18.718483,3.000000 58.029470,7.936083,18.771053,7.850386,3.486047,9.073247,6.786294,4.949442,3.565229,34.004830,84.326025,32.755850,41.777266,74.807435,32.737475,3.000000 47.655365,6.263117,10.347858,4.807681,4.465829,8.093180,5.107651,1.721795,4.607089,33.309515,102.655187,23.337544,38.665966,107.032195,18.660136,3.000000 37.110564,9.127239,10.668107,9.388768,7.970887,9.143346,12.781739,13.336964,6.699117,2.049043,44.085116,10.394698,5.635468,36.843559,23.718730,3.000000 38.684006,11.977840,10.109062,4.396552,1.626858,0.703276,12.755652,4.171569,4.843316,33.498154,83.193397,31.964912,32.269514,74.880485,33.148861,3.000000 37.850077,11.219031,11.298609,4.051946,5.171304,1.238764,9.662268,7.351379,3.868443,29.601931,85.170091,34.670269,24.091245,67.618521,43.369632,3.000000 33.375924,45.897643,44.959032,4.206236,2.510320,3.222605,6.785700,5.948434,5.084050,4.488179,10.262063,2.284405,10.728425,9.843519,2.926731,4.000000 42.745151,37.643024,50.160862,5.435130,4.830251,6.826986,5.556171,3.002356,0.205582,4.943730,4.228434,1.315191,14.777243,1.532173,2.396397,4.000000 36.649501,39.702291,44.822909,3.642930,8.181224,8.751886,3.456824,9.219551,4.474447,5.561067,16.236727,1.138055,18.158290,4.719707,7.470625,4.000000 33.732343,45.422586,55.822462,7.638146,2.152629,4.033472,1.464054,4.268573,1.663308,2.984584,7.626433,3.389159,6.941995,4.292772,5.763352,4.000000 38.116837,42.441980,54.622501,5.664158,2.892497,5.785950,4.736157,4.173156,2.245456,4.465982,7.503835,2.509713,12.375703,4.960036,2.874021,4.000000 24.033897,40.088233,38.787525,9.386012,9.464247,10.413310,5.538884,8.901695,6.428157,5.607933,8.132257,8.182374,12.767122,4.234681,9.934021,4.000000 22.047348,48.635157,48.085973,4.204146,8.782940,8.546689,8.453367,8.064216,4.190607,8.495972,9.646204,6.860789,15.861648,9.336959,7.072326,4.000000 33.365355,38.958717,46.813890,9.899286,7.299461,2.715544,9.758196,11.715413,5.594981,17.783596,9.339260,5.354658,18.192992,8.105073,12.362090,4.000000 38.594702,38.014552,39.186073,10.419738,4.358816,1.884457,3.774121,6.272337,6.311814,4.514871,3.737852,3.404841,16.175068,7.367417,6.986737,4.000000 34.595418,45.275342,38.231041,16.250354,8.341614,3.207451,15.948974,17.222175,0.766148,21.815026,15.449771,4.494622,21.761233,9.026763,6.234337,4.000000 35.255824,12.681438,12.162714,1.949913,4.178936,2.694380,5.003759,5.714465,4.993314,2.866081,7.437629,5.021466,6.839853,3.950859,7.966758,5.000000 35.381951,6.525500,19.199896,3.555485,5.347654,2.759508,1.414179,4.648061,6.142657,1.858605,7.096097,7.531731,6.483093,6.014813,12.057738,5.000000 41.645858,15.220567,8.432321,3.891638,2.995404,3.117573,2.547617,4.607339,4.529059,4.610331,8.582590,3.758644,12.137491,3.805865,7.507453,5.000000 31.975783,6.425621,27.088570,1.623824,0.324600,3.288059,4.708754,6.496409,2.555214,3.167269,10.568400,3.977490,5.989966,12.268311,8.626645,5.000000 52.354737,15.120519,8.159867,12.150277,3.072984,2.300643,6.897060,7.296752,5.010795,13.108232,14.797280,6.259755,33.110258,40.388509,18.435610,5.000000 42.999924,9.240689,12.167061,6.017784,5.659045,3.776685,8.656432,8.790815,2.774698,9.226938,7.683244,2.427294,5.585492,5.957392,7.547471,5.000000 43.477665,14.105631,13.982119,10.789480,3.810610,5.423428,6.094256,6.136796,5.340599,8.386305,7.252753,1.856217,5.491802,10.913479,5.328268,5.000000 20.702271,10.081805,4.982003,23.877205,7.948928,1.551362,19.273723,11.337202,3.647189,20.742387,9.034537,1.285506,19.839281,4.202442,2.836296,5.000000 17.150237,8.940540,9.556369,14.012305,6.129931,4.739453,8.193098,13.911509,5.344650,6.554458,10.177747,5.336161,3.375271,4.698845,3.420790,5.000000 39.389713,1.117582,18.518180,2.084555,9.091669,5.572484,4.861925,9.557979,1.797316,8.672537,6.123965,4.182991,5.379867,3.624339,9.622453,5.000000 32.768108,9.503798,21.593121,70.120588,79.257446,19.285731,62.786166,89.545887,25.282514,35.479257,113.421803,22.441275,2.555735,2.533934,11.763745,6.000000 46.482827,7.663155,23.567131,63.902830,93.346809,22.482054,56.086617,100.846042,23.564547,33.851183,117.178513,25.286268,6.500882,2.959386,19.501039,6.000000 38.634087,11.621667,12.321092,65.498904,85.213781,26.907686,64.852347,83.909753,32.034636,62.020283,83.323703,34.953183,29.379008,6.594299,10.074840,6.000000 36.741209,5.558563,16.938405,62.598507,83.631724,28.465062,62.411796,78.189144,32.007691,54.511854,86.981567,39.002972,30.600811,3.231690,22.656990,6.000000 32.129682,4.522638,20.253860,56.993430,92.831512,27.698204,64.360650,95.358514,25.051703,46.830764,114.904887,27.940107,11.182135,6.640856,21.113219,6.000000 44.091949,7.621931,10.947011,55.501159,89.272723,24.395429,55.948755,82.842522,30.297209,38.470464,104.984128,27.955863,14.350666,8.625918,9.942533,6.000000 27.479936,4.097992,12.034439,54.818352,83.537471,23.730609,48.670413,87.563799,21.889500,49.545667,79.728856,24.308692,24.089367,8.214144,6.436579,6.000000 36.959546,8.284973,14.947872,68.824398,94.120399,21.986314,66.216127,86.226797,19.891681,68.870759,84.502687,33.513477,39.479300,6.941457,13.541512,6.000000 31.751109,4.022216,18.241775,50.880285,84.077726,26.394814,47.129386,86.697861,22.057838,45.542544,80.530217,21.078212,23.212825,9.845619,9.567523,6.000000 32.046934,2.906928,16.507149,65.617373,85.127539,28.785882,59.764329,87.419937,28.859106,55.605317,84.626147,30.414226,24.107689,8.410958,11.024586,6.000000 44.811824,2.247344,16.590034,3.907380,1.037199,5.722256,24.259901,79.244223,50.637637,27.387484,89.464871,38.621035,18.931667,12.572601,2.455508,7.000000 37.624495,9.248881,15.595617,13.943688,6.967265,3.055396,23.551784,87.673090,25.656546,26.030782,73.236715,36.542141,17.261648,14.773418,0.548913,7.000000 46.128234,4.915813,20.525325,18.394542,5.316863,7.119637,36.039645,100.389825,28.537262,32.345505,93.461557,34.594623,16.737188,8.445271,13.111714,7.000000 44.808399,4.501318,15.921841,5.676711,2.912810,3.308981,47.062820,107.593445,23.310176,44.997130,102.337257,28.679087,22.346223,1.190159,15.552526,7.000000 26.530469,6.008313,17.595035,18.261555,3.655714,3.716061,38.781065,99.707067,33.522438,39.233322,94.661400,30.749988,17.065474,7.716530,4.521263,7.000000 25.322479,18.037882,12.067315,32.241247,4.182720,10.241490,64.298837,85.322581,35.394987,66.325426,85.916786,50.186207,33.529981,4.814649,21.073403,7.000000 26.159149,11.137073,9.392574,32.497866,5.530923,21.209995,58.690552,88.162466,44.455593,66.220628,82.889458,53.075216,26.491866,8.719384,20.402972,7.000000 27.655743,8.826697,10.426987,24.228312,4.138157,1.587366,43.504149,89.435706,34.546310,49.337679,85.551153,39.430488,23.411189,3.340633,15.438018,7.000000 36.260453,2.525784,12.413987,6.889512,1.646308,7.154511,41.815285,94.048841,25.811301,32.851651,98.766322,29.309239,13.612372,3.147535,24.566801,7.000000 24.798043,8.936647,9.570077,23.075096,20.969004,7.107377,43.418623,85.495565,23.589483,45.492214,78.922022,23.059844,20.685546,13.108988,7.381683,7.000000 24.596646,29.736908,25.081947,8.884024,6.559019,0.456588,29.827859,102.927281,18.240074,12.725527,97.401429,17.777858,15.098036,16.796076,6.547921,8.000000 50.252621,28.077263,30.040298,22.374885,2.026989,4.306113,48.841783,103.821814,30.767418,54.614023,96.278374,33.056198,26.596671,9.641538,4.516223,8.000000 46.720347,25.202566,24.210076,17.260718,9.712490,3.675929,28.947328,75.844410,28.755709,30.145959,65.809682,40.230338,16.848285,4.840667,8.473947,8.000000 28.024332,40.119232,25.496783,13.016671,7.596354,14.071081,32.813221,113.327516,27.055776,24.094514,106.255872,24.239304,2.695157,12.629440,4.089145,8.000000 34.887251,22.980568,30.863599,3.881194,2.064988,2.772694,38.137402,103.151437,25.618216,29.328686,97.400949,24.351962,5.864652,4.536465,12.272375,8.000000 44.262576,27.125702,15.256883,20.713364,3.406373,9.884314,48.267375,88.476030,29.769608,53.448143,86.022664,36.539874,34.193978,5.617906,18.224099,8.000000 34.424282,20.819976,23.608225,11.864899,2.669081,5.032476,42.855215,93.195115,35.297608,43.077306,90.292816,43.513382,13.542847,2.543531,21.420034,8.000000 23.015253,23.751870,6.535196,3.763148,11.911140,3.379227,11.032025,87.060834,28.273261,6.145853,86.565476,29.938395,12.812652,48.514888,26.460021,8.000000 |

## Model Builder for .NET

This section presents automated machine learning in Visual Studio .NET 2022 Community Edition. You are not expected to install Visual Studio .NET. There are no exercises for this section. My main goal though is to introduce an alternate automated machine learning platform within another code framework. The demonstration that I present uses C# and Model Builder which is an add-on from Microsoft for Visual Studio .NET 2022 Community Edition.

The demonstration is based heavily on the steps presented at:

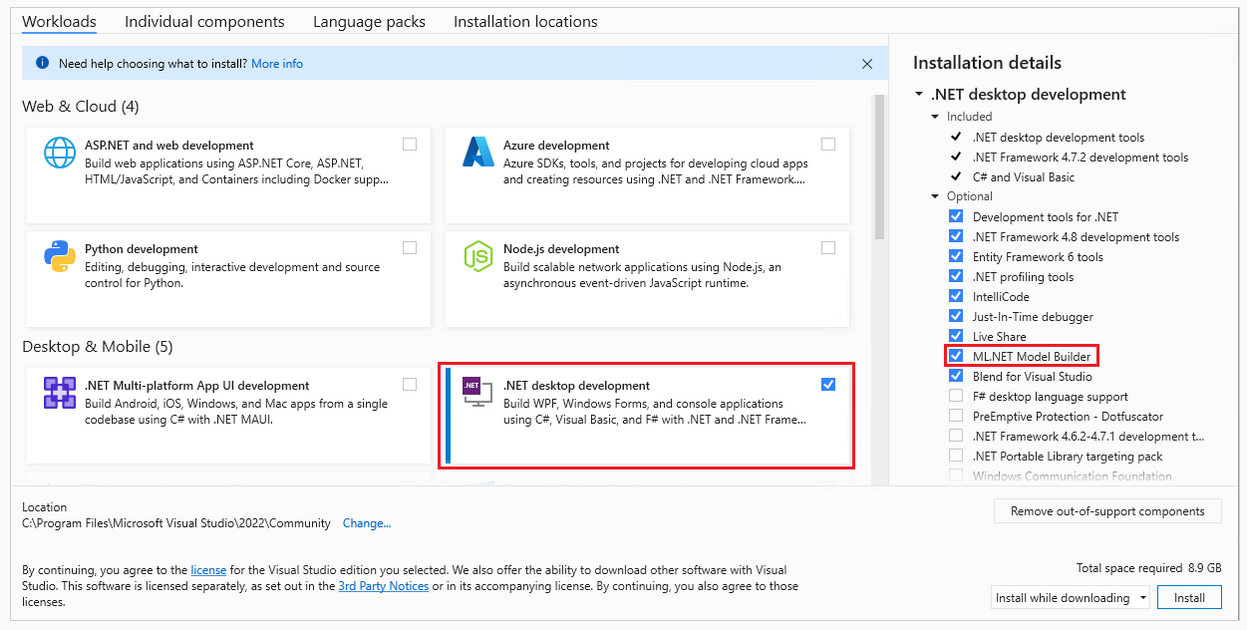
<https://dotnet.microsoft.com/en-us/learn/ml-dotnet/get-started-tutorial/create>

### Installation

Visual Studio .NET 2022 Community edition can be downloaded free from <https://visualstudio.microsoft.com/vs/community/>

In case you are not familiar with VS .NET Community it is an excellent environment and it offers basically everything you need for .NET and C# development for free.

During the install, when prompted choose .NET desktop development and if the option is given check ML.NET Model Builder.



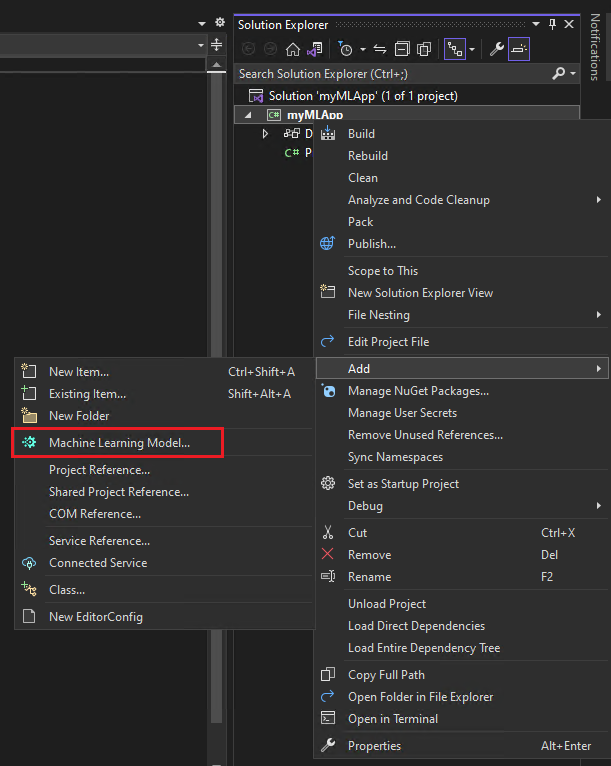
### Create the app

Open Visual Studio and create a new .NET console app:

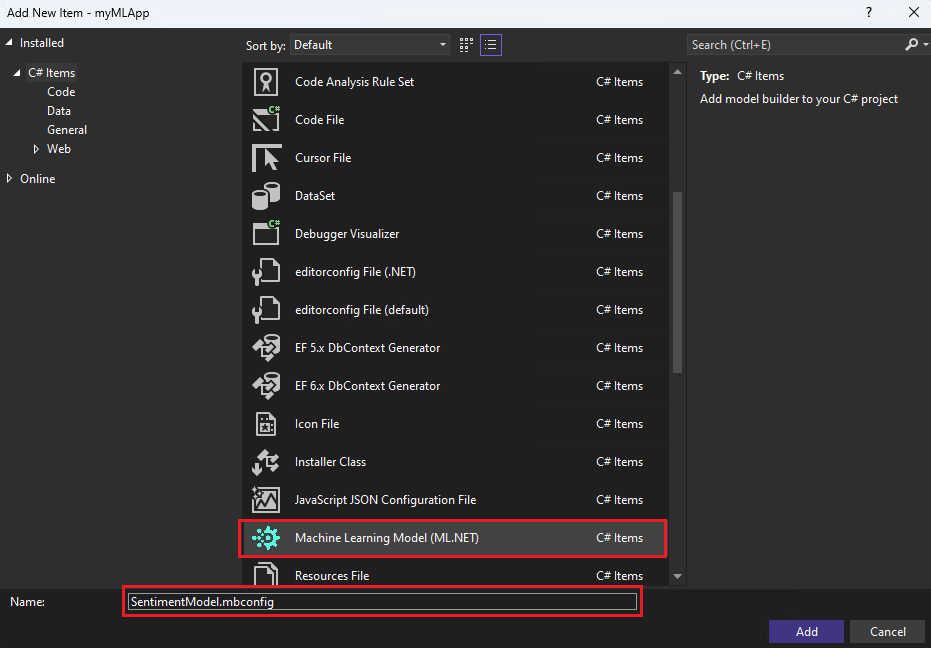
1. Select **Create a new project** from the Visual Studio 2022 start window.
2. Select the **C# Console App** project template.
3. Change the project name to **myMLApp**.
4. Make sure **Place solution and project in the same directory** is unchecked.
5. Select the **Next** button.
6. Select **.NET 7.0 (Standard Term support)** as the Framework.
7. Select the **Create** button. Visual Studio creates your project and loads the Program.cs file.

### Add machine learning

1. Right-click on the myMLApp project in **Solution Explorer** and select **Add** > **Machine Learning Model**.



1. In the **Add New Item** dialog, make sure **Machine Learning Model (ML.NET)** is selected.
2. Change the **Name** field to SentimentModel.mbconfig and select the **Add** button.

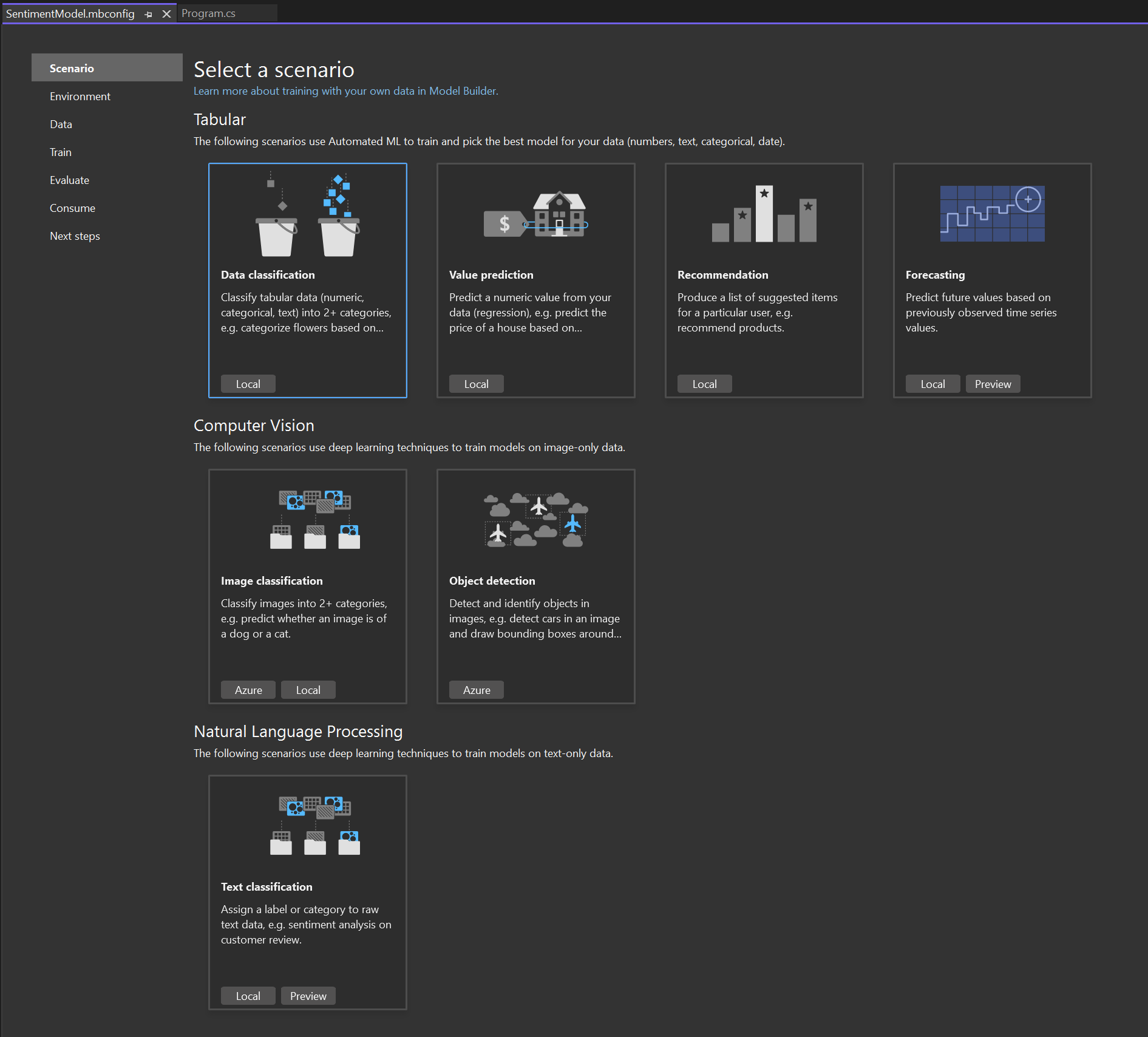


A new file named SentimentModel.mbconfig is added to your solution and the Model Builder UI opens in a new docked tool window in Visual Studio. The **mbconfig** file is simply a JSON file that keeps track of the state of the UI.

Model Builder will guide you through the process of building a machine learning model in the following steps.

### Pick a scenario

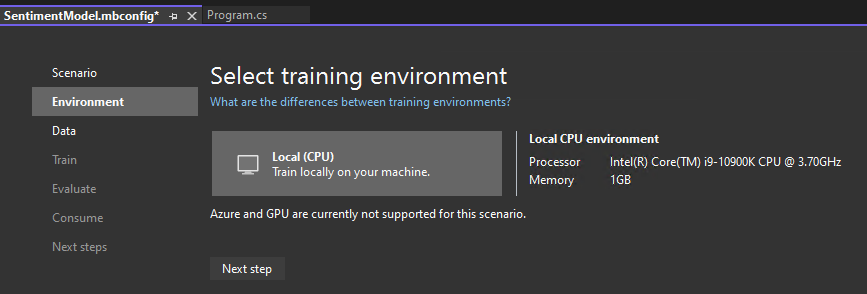
To generate your model, you first need to select your machine learning scenario. Model Builder supports several scenarios:



**Note:** If the tutorial screenshots don't match with what you see, you may need to update your version of Model Builder. Go to **Extensions** > **Manage Extensions** to make sure that there are no available updates for Model Builder. The version used in this tutorial is **16.14.0**.

In this case, you'll predict sentiment based on the content (text) of customer reviews.

1. In the Model Builder Scenario screen, select the **Data classification** scenario, since you're predicting which category a comment falls into (positive or negative).
2. After selecting the **Data classification** scenario, you must choose your training environment. While some scenarios support training in Azure, Classification currently only supports local training, so keep the **Local** environment selected and move on to the **Data** step.



### Download and add data

Download the [Sentiment Labelled Sentences datasets](https://archive.ics.uci.edu/ml/machine-learning-databases/00331/sentiment%20labelled%20sentences.zip) from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences). Unzip sentiment labelled sentences.zip and save the yelp\_labelled.txt file to the myMLApp directory.

Each row in yelp\_labelled.txt represents a different review of a restaurant left by a user on Yelp. The first column represents the comment left by the user, and the second column represents the sentiment of the text (0 is negative, 1 is positive). The columns are separated by tabs, and the dataset has no header. The data looks like the following:

yelp\_labelled.txt

Wow... Loved this place. 1

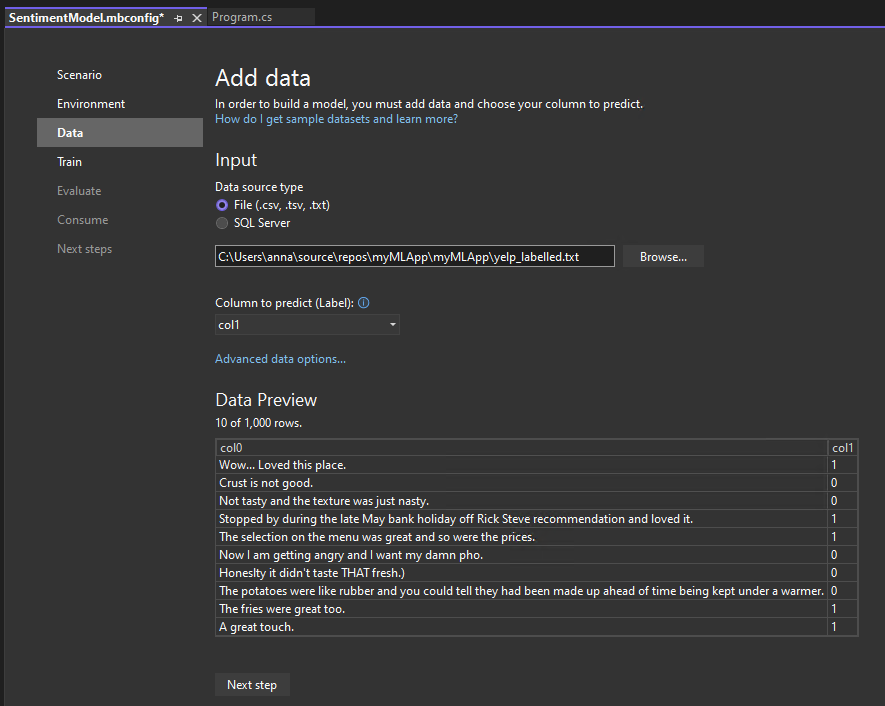
Crust is not good. 0

Not tasty and the texture was just nasty. 0

### Add data

In Model Builder, you can add data from a local file or connect to a SQL Server database. In this case, you'll add yelp\_labelled.txt from a file.

1. Select **File** as the input data source type.
2. Browse for yelp\_labelled.txt. Once you select your dataset, a preview of your data appears in the **Data Preview** section. Since your dataset does not have a header, headers are auto-generated ("col0" and "col1").
3. Under **Column to predict (Label)**, select "col1". The **Label** is what you're predicting, which in this case is the sentiment found in the second column ("col1") of the dataset.
4. The columns that are used to help predict the Label are called **Features**. All of the columns in the dataset besides the Label are automatically selected as Features. In this case, the review comment column ("col0") is the Feature column. You can update the Feature columns and modify other data loading options in **Advanced data options**, but it is not necessary for this example.



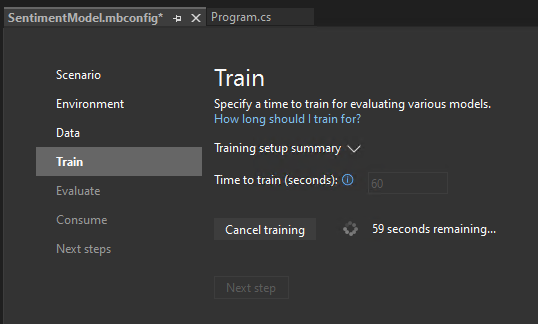
After adding your data, go to the **Train** step.

### Train your model

Now, you'll train your model with the yelp\_labelled.txt dataset.

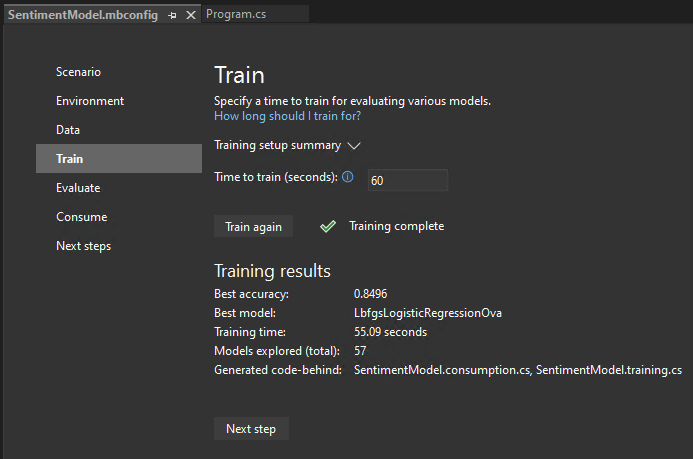
Model Builder evaluates many models with varying algorithms and settings based on the amount of training time given to build the best performing model.

1. Change the **Time to train**, which is the amount of time you'd like Model Builder to explore various models, to **60** seconds (you can try increasing this number if no models are found after training) . Note that for larger datasets, the training time will be longer. Model Builder automatically adjusts the training time based on the dataset size.
2. Select **Start training** to start the training process. Once training starts, you can see the time remaining.



### Training results

Once training is done, you can see a summary of the training results.



* **Best accuracy** - This shows you the accuracy of the best model that Model Builder found. Higher accuracy means the model predicted more correctly on test data.
* **Best model** - This shows you which algorithm performed the best during Model Builder's exploration.
* **Training time** - This shows you the total amount of time that was spent training / exploring models.
* **Models explored (total)** - This shows you the total number of models explored by Model Builder in the given amount of time.

If you want, you can view more information about the training session in the Machine Learning Output window.

After model training finishes, go to the **Evaluate** step.

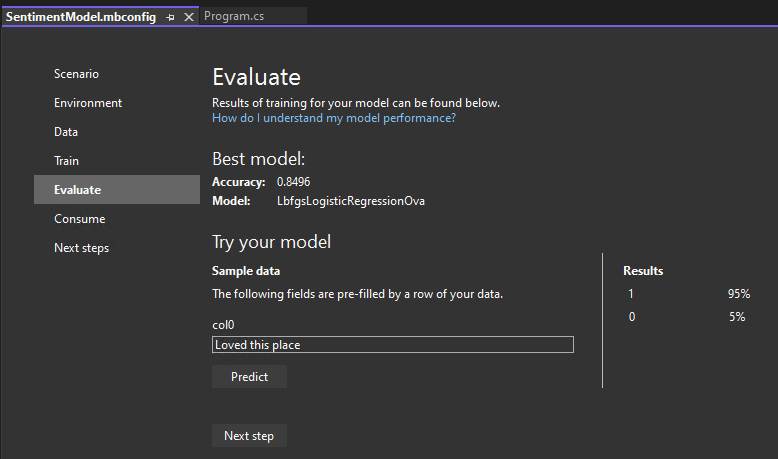
### Evaluate your model

The **Evaluate** step shows you the best-performing [algorithm](https://learn.microsoft.com/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm?WT.mc_id=dotnet-35129-website) and the best accuracy and lets you try out the model in the UI.

### Try out your model

You can make predictions on sample input in the **Try your model** section. The textbox is pre-filled with the first line of data from your dataset, but you can change the input and select the **Predict** button to try out different sentiment predictions.

In this case, 0 means negative sentiment and 1 means positive sentiment.



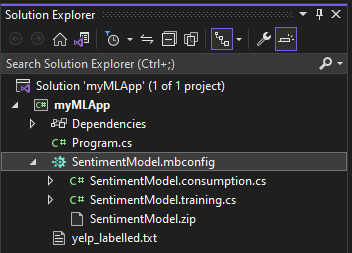
**Note:** If your model is not performing well (for example, if the Accuracy is low or if the model only predicts '1' values), you can try adding more time and training again. This is a sample using a very small dataset; for production-level models, you'd want to add a lot more data and training time.

After evaluating and trying out your model, move on to the **Consume** step.

### Generate code

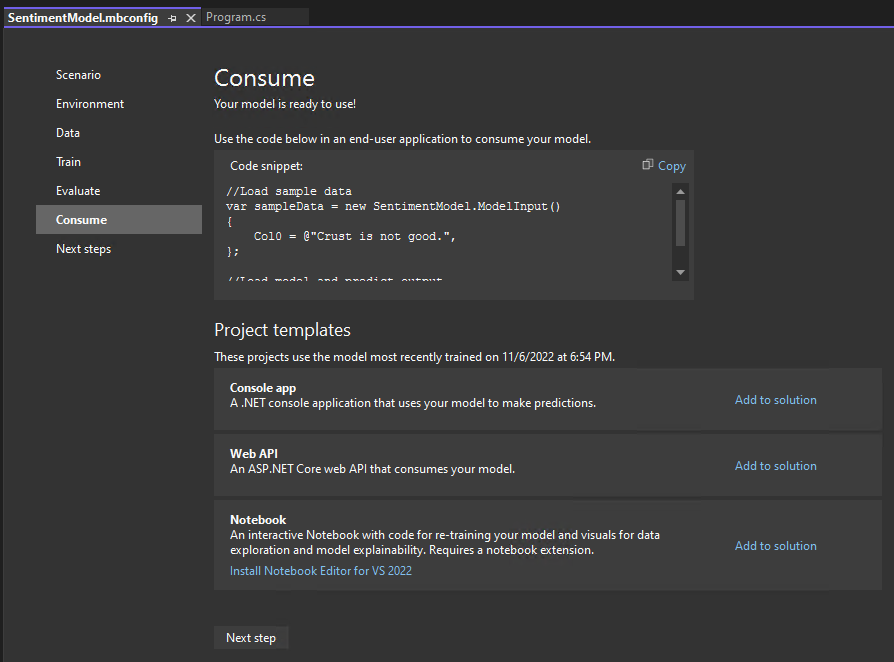
After training is completed, three files are automatically added as code-behind to the SentimentModel.mbconfig:

* SentimentModel.zip: This file is the trained ML.NET model, which is a serialized zip file.
* SentimentModel.consumption.cs: This file contains the model input and output classes and a Predict method that can be used for model consumption.
* SentimentModel.training.cs: This file contains the training pipeline (data transforms, algorithm, and algorithm parameters) used to train the final model.



In the **Consume** step in Model Builder, a code snippet is provided which creates sample input for the model and uses the model to make a prediction on that input.

Model Builder also offers **Project templates** that you can optionally add to your solution. There are two project templates (a console app and a web API), both which consume the trained model.



### Consume your model

The last step is to consume your trained model in the end-user application.

1. Replace the Program.cs code in your myMLApp project with the following code:

Program.cs

 using MyMLApp;

// Add input data

var sampleData = new SentimentModel.ModelInput()

{

Col0 = "This restaurant was wonderful."

};

// Load model and predict output of sample data

var result = SentimentModel.Predict(sampleData);

// If Prediction is 1, sentiment is "Positive"; otherwise, sentiment is "Negative"

var sentiment = result.PredictedLabel == 1 ? "Positive" : "Negative";

Console.WriteLine($"Text: {sampleData.Col0}\nSentiment: {sentiment}");

 Run myMLApp (select **Ctrl+F5** or **Debug** > **Start Without Debugging**). You should see the following output, predicting whether the input statement is positive or negative.

