Machine Learning

Kelompok J

11:11PM

Member

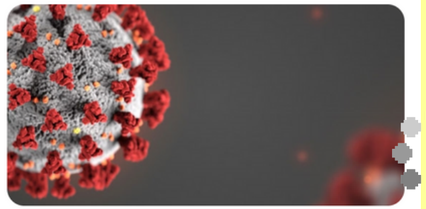
Antonius Kevin B S (00000045444)
Chrealvin (00000045606)
Ela Setiorini (00000048258)
Muhammad Farrel P (00000051515)

Back to Agenda Page

Dataset

COVID-19 Stats and Mobility Trends

Population Mobility Trends, Country-specific Indicators



Analyze and Predict Covid-19 Data From Domain Status and Mobility

Number of Data : >3000 sample

Back to Agenda Page

Data Preprocessing

Select only Southeast Asia data

```
np.unique(covid['country'])
```
[4] ✓ 0.2s

```
array(['Afghanistan', 'Angola', 'Antigua and Barbuda', 'Argentina',
       'Aruba', 'Australia', 'Austria', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bolivia',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil', 'Bulgaria',
       'Burkina Faso', 'Cambodia', 'Cameroon', 'Canada', 'Cape Verde',
       'Chile', 'Colombia', 'Costa Rica', 'Croatia', 'Czechia',
       "Côte d'Ivoire", 'Denmark', 'Dominican Republic', 'Ecuador',
       'Egypt', 'El Salvador', 'Estonia', 'Fiji', 'Finland', 'France',
       'Gabon', 'Georgia', 'Germany', 'Ghana', 'Greece', 'Guatemala',
       'Guinea-Bissau', 'Haiti', 'Honduras', 'Hong Kong', 'Hungary',
       'India', 'Indonesia', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kuwait',
       'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Libya',
       'Liechtenstein', 'Lithuania', 'Luxembourg', 'Malaysia', 'Mali',
       'Malta', 'Mauritius', 'Mexico', 'Moldova', 'Mongolia', 'Morocco',
       'Mozambique', 'Namibia', 'Nepal', 'Netherlands', 'New Zealand',
       'Nicaragua', 'Niger', 'Nigeria', 'North Macedonia', 'Norway',
       'Oman', 'Pakistan', 'Panama', 'Papua New Guinea', 'Paraguay',
       'Peru', 'Philippines', 'Poland', 'Portugal', 'Puerto Rico',
       'Qatar', 'Romania', 'Russia', 'Rwanda', 'Réunion', 'Saudi Arabia',
       'Senegal', 'Serbia', 'Singapore', 'Slovakia', 'Slovenia',
       'South Africa', 'South Korea', 'Spain', 'Sri Lanka', 'Sweden',
       'Switzerland', 'Taiwan', 'Tajikistan', 'Tanzania', 'Thailand',
       'Togo', 'Trinidad and Tobago', 'Turkey', 'Uganda', 'Ukraine',
       'United Arab Emirates', 'United Kingdom', 'United States',
       'Uruguay', 'Venezuela', 'Vietnam', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype=object)
```

```
covid_ind = covid[covid['country']=='Indonesia']
covid_mly = covid[covid['country']=='Malaysia']
covid_sg = covid[covid['country']=='Singapore']
covid_th = covid[covid['country']=='Thailand']
covid_vt = covid[covid['country']=='Vietnam']
covid_laos = covid[covid['country']=='Laos']
covid_ph = covid[covid['country']=='Philipines']
covid_cam = covid[covid['country']=='Cambodia']
```
[5] ✓ 0.6s

```
data = pd.concat([covid_ind, covid_mly, covid_sg, covid_th, covid_vt, covid_laos, covid_ph, covid_cam])
```
[6] ✓ 0.2s

```
np.unique(data['country'])
```
[7] ✓ 0.6s

```
array(['Cambodia', 'Indonesia', 'Laos', 'Malaysia', 'Singapore',
       'Thailand', 'Vietnam'], dtype=object)
```

Back to Agenda Page

Check correlation between data

```
[10]  covid.corr()
```

Python

	grocery_and_pharmacy_percent_change_from_baseline	parks_percent_change_from_baseline	residential_percent_change_from_
grocery_and_pharmacy_percent_change_from_baseline	1.000000	0.507787	-(
parks_percent_change_from_baseline	0.507787	1.000000	-(
residential_percent_change_from_baseline	-0.577338	-0.538879	
retail_and_recreation_percent_change_from_baseline	0.789658	0.724229	-(
transit_stations_percent_change_from_baseline	0.732437	0.611633	-(
workplaces_percent_change_from_baseline	0.477889	0.330226	-(
confirmed_cases	0.065004	0.021115	(
confirmed_deaths	0.056546	0.066716	-(
gov_response_stringency_index	-0.333530	-0.416354	(
total_tests	0.644449	0.094385	-(
total_vaccinations	0.032281	0.353975	(
people_vaccinated	0.026171	0.350912	(
people_fully_vaccinated	0.266336	0.425693	-(
gdp_ppp_per_capita	0.034540	-0.150927	(
population	0.092480	0.056395	-(
population_density	0.044633	-0.099650	(
human_development_index	0.115143	-0.214043	(
pop_age_above_65_percentage	0.235356	-0.170593	(
health_index	0.096514	-0.202410	(

Back to Agenda Page

```
[33]  covid.info()

...  <class 'pandas.core.frame.DataFrame'>
     DatetimeIndex: 2821 entries, 2020-02-15 to 2021-03-23
     Data columns (total 21 columns):
      #   Column                                              Non-Null Count  Dtype
     ---  ------                                              --------------  -----
      0   iso                                                 2821 non-null   object
      1   country                                             2821 non-null   object
      2   grocery_and_pharmacy_percent_change_from_baseline   2821 non-null   float64
      3   parks_percent_change_from_baseline                  2821 non-null   float64
      4   residential_percent_change_from_baseline            2821 non-null   float64
      5   retail_and_recreation_percent_change_from_baseline  2821 non-null   float64
      6   transit_stations_percent_change_from_baseline       2821 non-null   float64
      7   workplaces_percent_change_from_baseline             2821 non-null   float64
      8   confirmed_cases                                     2821 non-null   int64
      9   confirmed_deaths                                    2821 non-null   int64
      10  gov_response_stringency_index                       2816 non-null   float64
      11  total_tests                                         401 non-null    float64
      12  total_vaccinations                                  136 non-null    float64
      13  people_vaccinated                                   136 non-null    float64
      14  people_fully_vaccinated                             59 non-null     float64
      15  gdp_ppp_per_capita                                  2821 non-null   float64
      16  population                                          2821 non-null   float64
      17  population_density                                  2821 non-null   float64
      18  human_development_index                             2821 non-null   float64
      19  pop_age_above_65_percentage                         2821 non-null   float64
      20  health_index                                        2821 non-null   float64
     dtypes: float64(17), int64(2), object(2)
     memory usage: 484.9+ KB
```

View all columns in dataset

Back to Agenda Page

Drop unused data

```
[35]
covid.drop(columns=['iso', 'country',
       'gov_response_stringency_index', 'people_fully_vaccinated', 'population',
       'gdp_ppp_per_capita', 'population_density',
       'human_development_index', 'pop_age_above_65_percentage',
       'health_index'], inplace=True)
```

```
[36]
covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2821 entries, 2020-02-15 to 2021-03-23
Data columns (total 11 columns):
 #   Column                                              Non-Null Count  Dtype
---  ------                                              --------------  -----
 0   grocery_and_pharmacy_percent_change_from_baseline   2821 non-null   float64
 1   parks_percent_change_from_baseline                  2821 non-null   float64
 2   residential_percent_change_from_baseline            2821 non-null   float64
 3   retail_and_recreation_percent_change_from_baseline  2821 non-null   float64
 4   transit_stations_percent_change_from_baseline       2821 non-null   float64
 5   workplaces_percent_change_from_baseline             2821 non-null   float64
 6   confirmed_cases                                     2821 non-null   int64
 7   confirmed_deaths                                    2821 non-null   int64
 8   total_tests                                         401 non-null    float64
 9   total_vaccinations                                  136 non-null    float64
 10  people_vaccinated                                   136 non-null    float64
dtypes: float64(9), int64(2)
memory usage: 264.5 KB
```

Back to Agenda Page

Drop NaN from Total Test

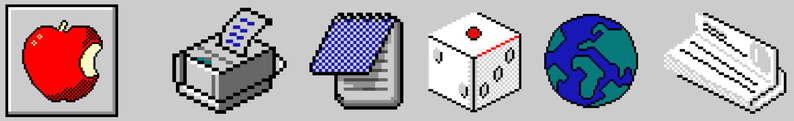
```
[16]
covid.dropna(subset=['total_tests'], inplace=True)
```
Python

```
[17]
covid.corr()
```
Python

...

	grocery_and_pharmacy_percent_change_from_baseline	parks_percent_change_from_baseline	residential_percent_change_from_baseline	retail_and_recreatior
grocery_and_pharmacy_percent_change_from_baseline	1.000000	0.584324	-0.674230	
parks_percent_change_from_baseline	0.584324	1.000000	-0.541261	
residential_percent_change_from_baseline	-0.674230	-0.541261	1.000000	
retail_and_recreation_percent_change_from_baseline	0.799249	0.807661	-0.865762	
transit_stations_percent_change_from_baseline	0.456517	0.714959	-0.713738	
workplaces_percent_change_from_baseline	0.306469	0.104093	-0.737159	
confirmed_cases	0.589723	-0.052224	-0.413028	
confirmed_deaths	0.494550	-0.019829	-0.318376	
total_tests	0.644449	0.094385	-0.593422	
total_vaccinations	-0.692912	-0.351817	0.438048	
people_vaccinated	-0.692912	-0.351817	0.438048	

```
[18]
covid.fillna(0, inplace=True)
```
Python



Back to Agenda Page

Check data correlation again

```
[19]  covid.corr()
```

Python

	grocery_and_pharmacy_percent_change_from_baseline	parks_percent_change_from_baseline	residential_percent_change_from_baseline	retail_and_recreatior
grocery_and_pharmacy_percent_change_from_baseline	1.000000	0.584324	-0.674230	
parks_percent_change_from_baseline	0.584324	1.000000	-0.541261	
residential_percent_change_from_baseline	-0.674230	-0.541261	1.000000	
retail_and_recreation_percent_change_from_baseline	0.799249	0.807661	-0.865762	
transit_stations_percent_change_from_baseline	0.456517	0.714959	-0.713738	
workplaces_percent_change_from_baseline	0.306469	0.104093	-0.737159	
confirmed_cases	0.589723	-0.052224	-0.413028	
confirmed_deaths	0.494550	-0.019829	-0.318376	
total_tests	0.644449	0.094385	-0.593422	
total_vaccinations	0.128056	0.006824	-0.091492	
people_vaccinated	0.128056	0.006824	-0.091492	

Back to Agenda Page

```
[23]  moved_column = covid.pop("confirmed_cases")
                                                                    Python

[24]  covid.insert(10, "confirmed_cases", moved_column)
                                                                    Python

[25]  covid.head()
                                                                    Python
```

...

date	grocery_and_pharmacy_percent_change_from_baseline	parks_percent_change_from_baseline	residential_percent_change_from_baseline	retail_and_recreation_percent_change_from_baseline	transit_stati
2020-02-15	1.0	-8.0	1.0	-3.0	
2020-02-16	2.0	-5.0	2.0	-3.0	
2020-02-17	1.0	-3.0	1.0	-3.0	
2020-02-18	0.0	-5.0	2.0	-3.0	
2020-02-19	-3.0	-3.0	1.0	-4.0	

Move 'confirmed_cases' to last column

Collect Covid OF Data as Values

```
[26]  covid_data = covid.values
```

```
[27]  covid_data
```

```
array([[ 1.0000e+00, -8.0000e+00,  1.0000e+00, ...,  0.0000e+00,
         0.0000e+00,  3.4000e+01],
       [ 2.0000e+00, -5.0000e+00,  2.0000e+00, ...,  0.0000e+00,
         0.0000e+00,  3.4000e+01],
       [ 1.0000e+00, -3.0000e+00,  1.0000e+00, ...,  0.0000e+00,
         0.0000e+00,  3.5000e+01],
       ...,
       [ 2.1000e+01, -1.9000e+01, -2.0000e+00, ...,  0.0000e+00,
         0.0000e+00,  2.7594e+04],
       [ 2.0000e+01, -2.0000e+01, -3.0000e+00, ...,  0.0000e+00,
         0.0000e+00,  2.7713e+04],
       [ 2.0000e+01, -1.2000e+01, -3.0000e+00, ...,  0.0000e+00,
         0.0000e+00,  2.7803e+04]])
```

```
[53]  X = covid_data[:,0:10]
```

```
[54]  X.shape
```

```
(401, 10)
```

```
[55]  Y = covid_data[:, 10]
```

```
[56]  Y
```

```
Output exceeds the size limit. Open the full output data in a text editor
array([   34.,    34.,    35.,    35.,    35.,    35.,    35.,    35.,
          35.,    35.,    37.,    40.,    40.,    40.,    40.,    42.,
          43.,    43.,    43.,    47.,    48.,    50.,    50.,    50.,
          53.,    59.,    70.,    75.,    82.,   114.,   147.,   177.,
         212.,   272.,   322.,   411.,   599.,   721.,   827.,   934.,
        1045.,  1136.,  1245.,  1388.,  1524.,  1651.,  1771.,  1875.,
        1978.,  2067.,  2169.,  2220.,  2258.,  2369.,  2423.,  2473.,
        2518.,  2551.,  2579.,  2613.,  2643.,  2672.,  2700.,  2733.,
        2765.,  2792.,  2811.,  2826.,  2839.,  2854.,  2907.,  2922.,
        2931.,  2938.,  2947.,  2954.,  2960.,  2966.,  2969.,  2987.,
        2988.,  2989.,  2992.,  3000.,  3004.,  3009.,  3015.,  3017.,
        3017.,  3018.,  3025.,  3026.,  3028.,  3031.,  3033.,  3034.,
        3037.,  3038.,  3040.,  3041.,  3042.,  3045.,  3054.,  3065.,
        3076.,  3077.,  3081.,  3082.,  3083.,  3084.,  3101.,  3102.,
        3104.,  3112.,  3119.,  3121.,  3125.,  3127.,  3129.,  3134.,
        3135.,  3136.,  3137.,  3139.,  3141.,  3146.,  3147.,  3148.,
        3151.,  3156.,  3157.,  3158.,  3162.,  3164.,  3166.,  3169.,
        3171.,  3173.,  3179.,  3180.,  3185.,  3190.,  3195.,  3196.,
        3197.,  3202.,  3209.,  3216.,  3217.,  3220.,  3227.,  3232.,
        3236.,  3239.,  3246.,  3249.,  3250.,  3255.,  3261.,  3269.,
        3279.,  3282.,  3291.,  3295.,  3297.,  3298.,  3304.,  3310.,
        3312.,  3317.,  3320.,  3321.,  3328.,  3330.,  3345.,  3348.,
        3351.,  3352.,  3354.,  3356.,  3359.,  3376.,  3376.,  3377.,
```

```
[60]  np.max(Y)
```

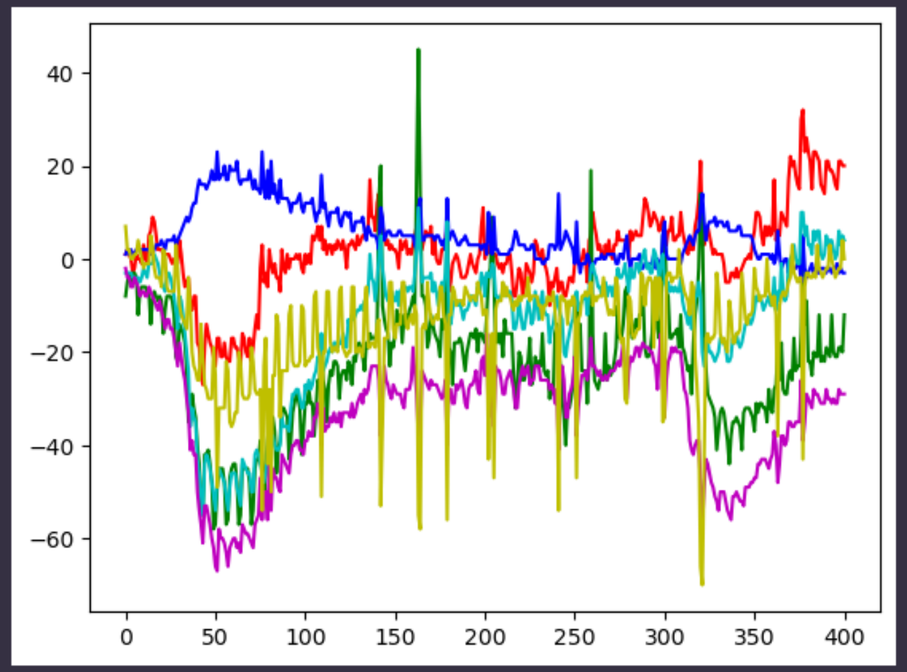
```
27803.0
```

Plot to see data scatter

```
plt.plot(X[:,0], color="r", label="grocery")
plt.plot(X[:,1], color="g", label="park")
plt.plot(X[:,2], color="b", label="residential")
plt.plot(X[:,3], color="c", label="retail")
plt.plot(X[:,4], color="m", label="transit")
plt.plot(X[:,5], color="y", label="workplace")
plt.show()
```

[82]

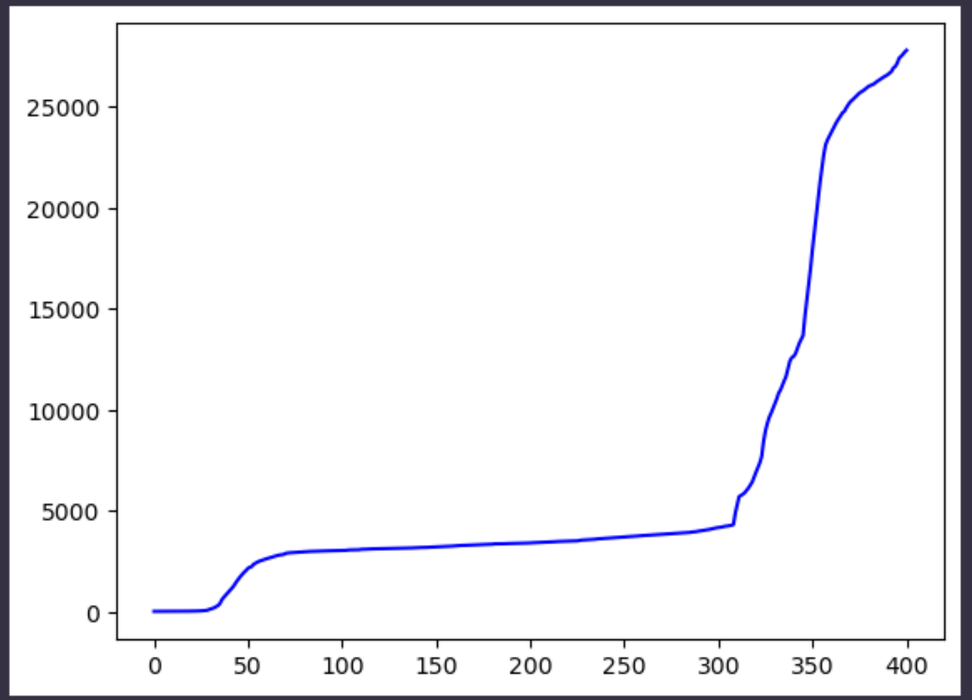
...




```
plt.plot(Y, color="b", label='Confirmed_case')
plt.show()
```

[87]

...



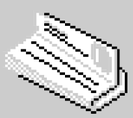
Back to Agenda Page

```
[88]
mms = MinMaxScaler(feature_range=(0,1))
scaled = mms.fit_transform(covid_data)

[89]
X_scaled = scaled[:,0:10]

[90]
Y_scaled = scaled[:,10]
```

Scale data with MinMaxScaler & Divide data to X and Y variable



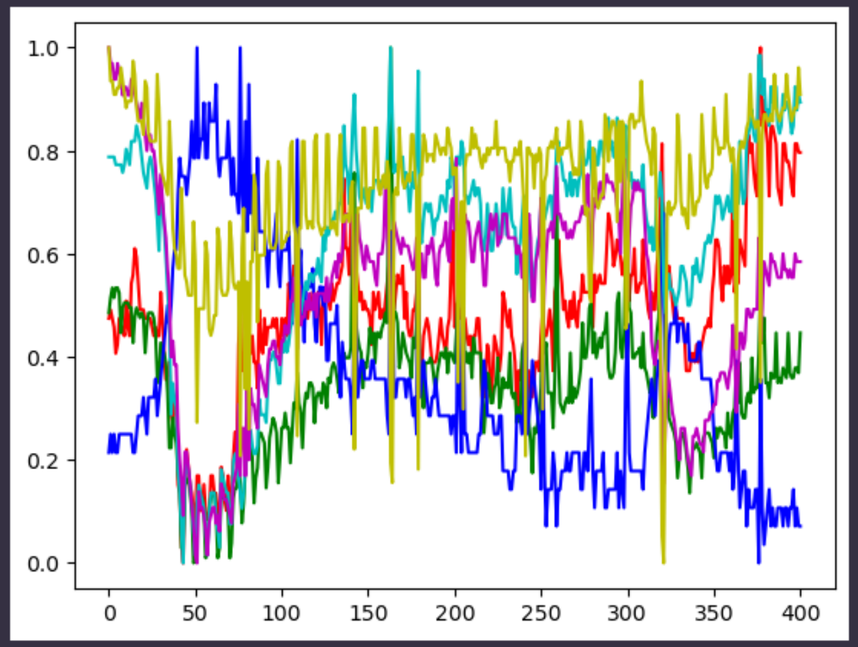
Back to Agenda Page

Plot to see scaled data spread

```
plt.plot(X_scaled[:,0], color="r", label="grocery")
plt.plot(X_scaled[:,1], color="g", label="park")
plt.plot(X_scaled[:,2], color="b", label="residential")
plt.plot(X_scaled[:,3], color="c", label="retail")
plt.plot(X_scaled[:,4], color="m", label="transit")
plt.plot(X_scaled[:,5], color="y", label="workplace")
plt.show()
```

[95]

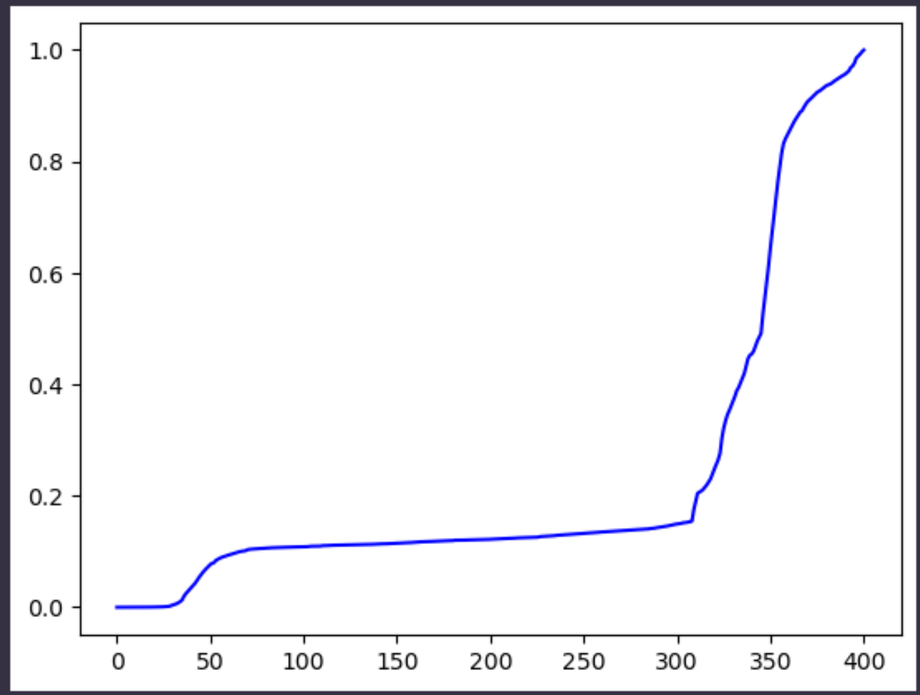
...




```
plt.plot(Y_scaled, color="b", label='Confirmed_case')
plt.show()
```

[93]

...



Back to Agenda Page

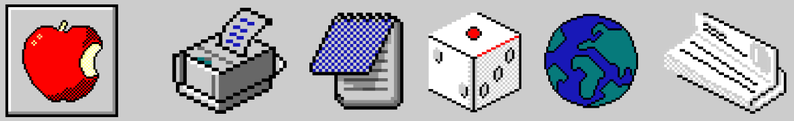
Splitting Data to 80% Training Data, 10% Test Data, 10% Validation Data

```
[96]  X_train, X_vt, Y_train, Y_vt = train_test_split(X_scaled, Y_scaled, test_size=0.2)

[97]  X_test, X_val, Y_test, Y_val = train_test_split(X_vt, Y_vt, test_size=0.5)

[98]  print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)

...   (320, 10) (41, 10) (40, 10) (320,) (41,) (40,)
```



Back to Agenda Page

Model initialization & compile model

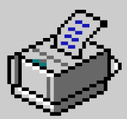
```
model = Sequential([
    Dense(256, activation='relu', input_shape=(10,)),
    Dropout(0.4),
    Dense(256, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

[44] Python

```
2022-12-06 12:34:51.010904: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

[45]

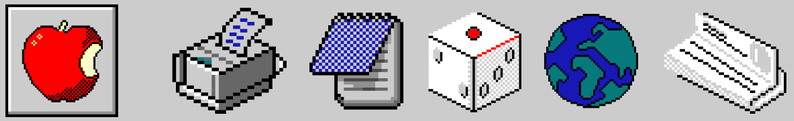


Back to Agenda Page

Start training and fitting the model

```
[46] hist = model.fit(X_train, Y_train,
                      batch_size=128, epochs=100,
                      validation_data=(X_val, Y_val))
```

```
... Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
3/3 [==============================] - 0s 55ms/step - loss: 0.6705 - accuracy: 0.0000e+00 - val_loss: 0.6334 - val_accuracy: 0.0000e+00
Epoch 2/100
3/3 [==============================] - 0s 10ms/step - loss: 0.5876 - accuracy: 0.0036 - val_loss: 0.6107 - val_accuracy: 0.0000e+00
Epoch 3/100
3/3 [==============================] - 0s 11ms/step - loss: 0.5383 - accuracy: 0.0036 - val_loss: 0.6171 - val_accuracy: 0.0000e+00
Epoch 4/100
3/3 [==============================] - 0s 10ms/step - loss: 0.5225 - accuracy: 0.0036 - val_loss: 0.6171 - val_accuracy: 0.0000e+00
Epoch 5/100
3/3 [==============================] - 0s 10ms/step - loss: 0.5155 - accuracy: 0.0036 - val_loss: 0.5790 - val_accuracy: 0.0000e+00
Epoch 6/100
3/3 [==============================] - 0s 10ms/step - loss: 0.4898 - accuracy: 0.0036 - val_loss: 0.5224 - val_accuracy: 0.0000e+00
Epoch 7/100
3/3 [==============================] - 0s 10ms/step - loss: 0.4652 - accuracy: 0.0036 - val_loss: 0.4682 - val_accuracy: 0.0000e+00
Epoch 8/100
3/3 [==============================] - 0s 10ms/step - loss: 0.4439 - accuracy: 0.0036 - val_loss: 0.4313 - val_accuracy: 0.0000e+00
Epoch 9/100
3/3 [==============================] - 0s 10ms/step - loss: 0.4235 - accuracy: 0.0071 - val_loss: 0.4034 - val_accuracy: 0.0000e+00
Epoch 10/100
3/3 [==============================] - 0s 10ms/step - loss: 0.4082 - accuracy: 0.0071 - val_loss: 0.3845 - val_accuracy: 0.0000e+00
Epoch 11/100
3/3 [==============================] - 0s 10ms/step - loss: 0.3931 - accuracy: 0.0071 - val_loss: 0.3765 - val_accuracy: 0.0000e+00
Epoch 12/100
3/3 [==============================] - 0s 10ms/step - loss: 0.3879 - accuracy: 0.0071 - val_loss: 0.3645 - val_accuracy: 0.0000e+00
Epoch 13/100
...
Epoch 99/100
3/3 [==============================] - 0s 10ms/step - loss: 0.3610 - accuracy: 0.0071 - val_loss: 0.3328 - val_accuracy: 0.0000e+00
Epoch 100/100
3/3 [==============================] - 0s 11ms/step - loss: 0.3608 - accuracy: 0.0071 - val_loss: 0.3335 - val_accuracy: 0.0000e+00
```

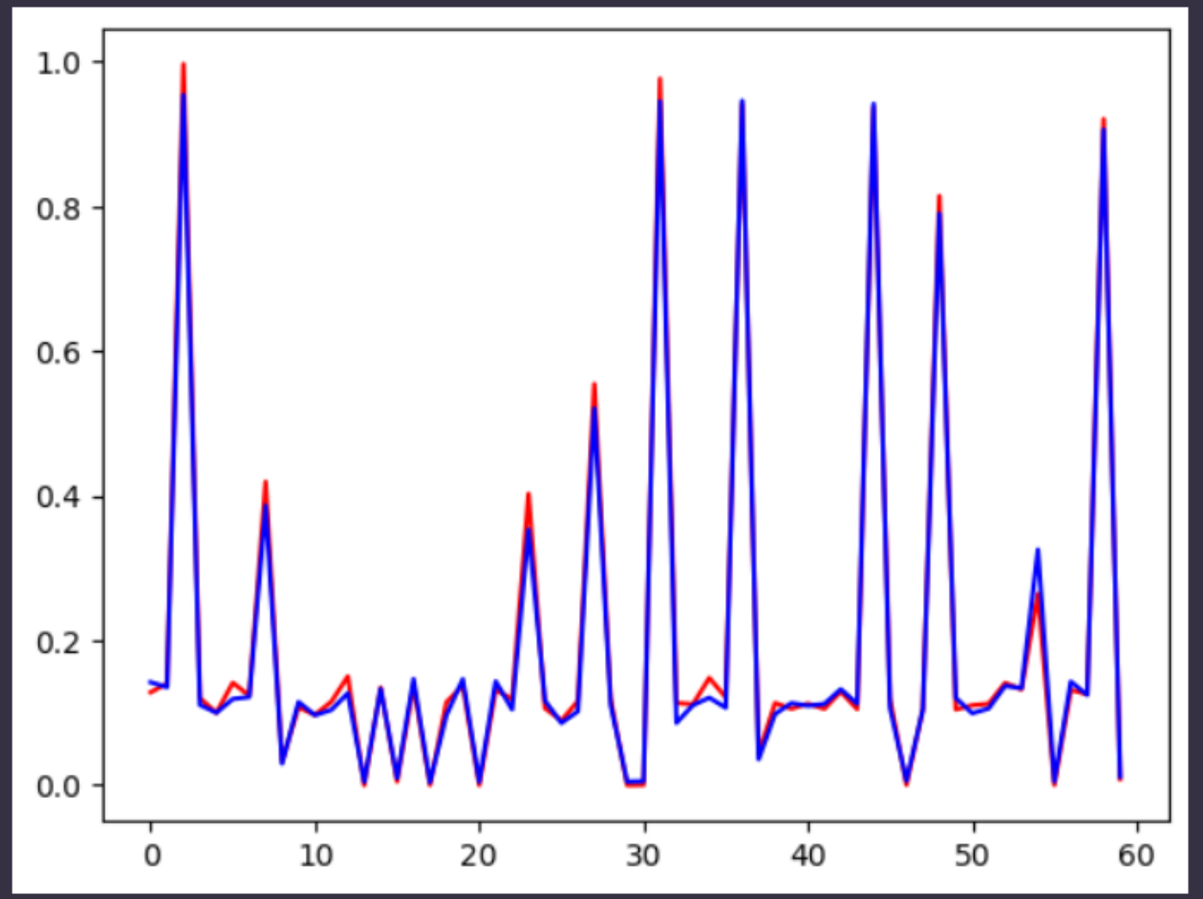


Back to Agenda Page

Plot model to see model movement

```
[50]
    plt.plot(Y_test, color='r', label="RealData")
    plt.plot(test, color='b', label='Predicted Test')
    plt.show()
```

...



Back to Agenda Page

Convert from scaled data to Real Value for Prediction Result and Real Data (X_test, Y_test)

```
[67]
test_value = []
for t in test:
    test_value.append(t*np.max(Y))
test_value[:10]
```

```
[array([4189.589], dtype=float32),
 array([3993.3684], dtype=float32),
 array([25411.213], dtype=float32),
 array([2953.7124], dtype=float32),
 array([2951.6702], dtype=float32),
 array([3634.5989], dtype=float32),
 array([3447.0413], dtype=float32),
 array([10990.064], dtype=float32),
 array([1068.5333], dtype=float32),
 array([3405.321], dtype=float32)]
```

```
[66]
Y_test_value = []
for y in Y_test:
    Y_test_value.append(y*np.max(Y))
Y_test_value[:10]
```

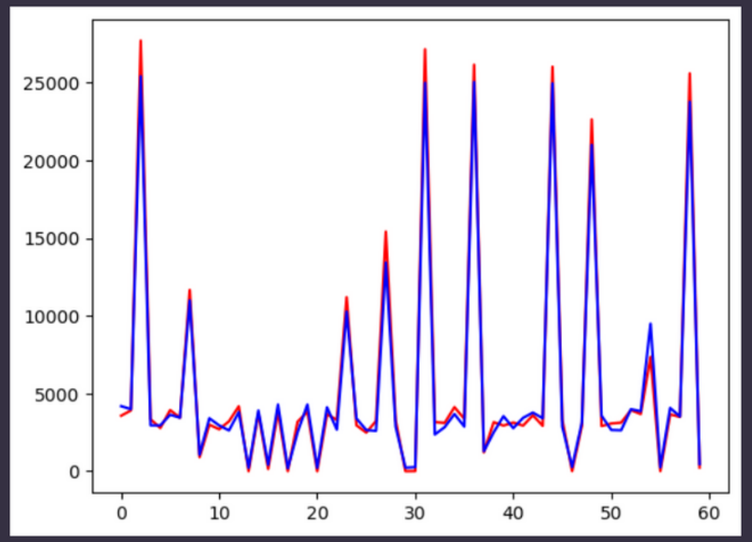
```
[3570.3661637077307,
 3912.7849040296724,
 27712.889805178434,
 3346.0919010407283,
 2780.4001224386902,
 3936.8142893154236,
 3443.210666570636,
 11660.259209910331,
 901.1019482156361,
 2994.6621412366308]
```

Back to Agenda Page

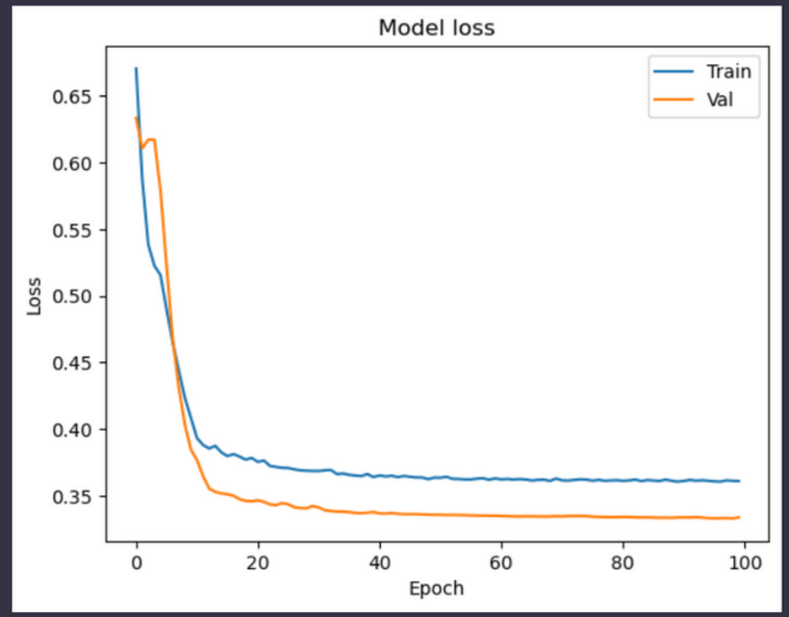
```
plt.plot(Y_test_value, color='r', label="RealData")
plt.plot(test_value, color='b', label='Predicted Test')
plt.show()
```

[63]

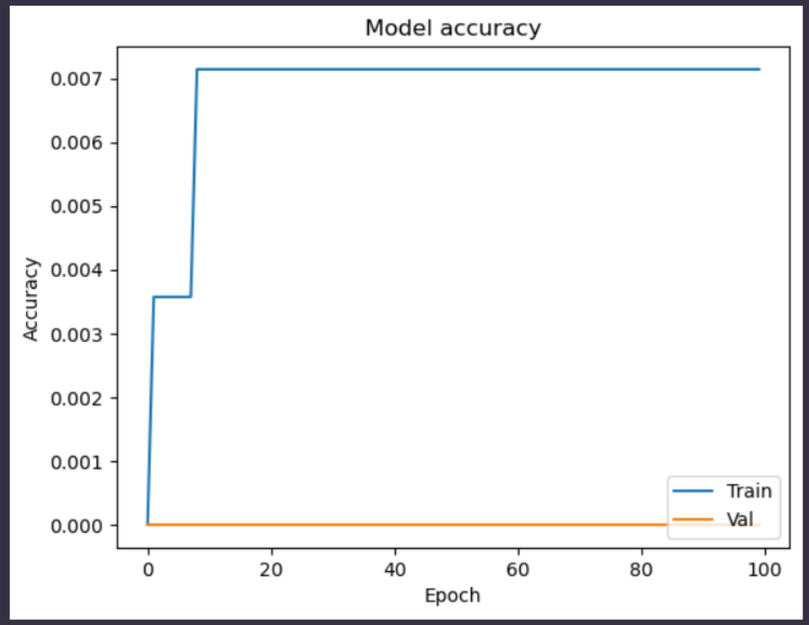
...




```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```




```
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```



Plot model based on data
Plot model loss &
Plot model accuracy

Train Model with different Input and Hidden Layer

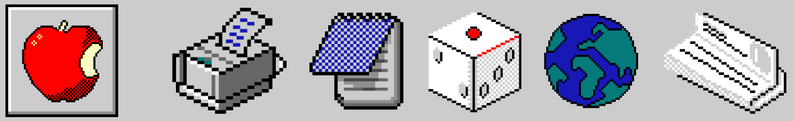
```
model_2 = Sequential([
    Dense(1000, activation='relu', input_shape=(10,)),
    Dropout(0.8),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1000, activation='relu'),
    Dense(1, activation='sigmoid'),
])

model_2.compile(optimizer='adam',
                loss='binary_crossentropy',
                metrics=['accuracy'])

hist_2 = model_2.fit(X_train, Y_train,
          batch_size=32, epochs=10,
          validation_data=(X_val, Y_val))
```

[53]

```
Epoch 1/10
9/9 [==============================] - 1s 22ms/step - loss: 0.5758 - accuracy: 0.0036 - val_loss: 0.5380 - val_accuracy: 0.0000e+00
Epoch 2/10
9/9 [==============================] - 0s 10ms/step - loss: 0.4763 - accuracy: 0.0036 - val_loss: 0.4280 - val_accuracy: 0.0000e+00
Epoch 3/10
9/9 [==============================] - 0s 10ms/step - loss: 0.4254 - accuracy: 0.0071 - val_loss: 0.4041 - val_accuracy: 0.0000e+00
Epoch 4/10
9/9 [==============================] - 0s 10ms/step - loss: 0.3982 - accuracy: 0.0071 - val_loss: 0.3662 - val_accuracy: 0.0000e+00
Epoch 5/10
9/9 [==============================] - 0s 10ms/step - loss: 0.3820 - accuracy: 0.0071 - val_loss: 0.3420 - val_accuracy: 0.0000e+00
Epoch 6/10
9/9 [==============================] - 0s 10ms/step - loss: 0.3759 - accuracy: 0.0071 - val_loss: 0.3386 - val_accuracy: 0.0000e+00
Epoch 7/10
9/9 [==============================] - 0s 13ms/step - loss: 0.3717 - accuracy: 0.0071 - val_loss: 0.3356 - val_accuracy: 0.0000e+00
Epoch 8/10
9/9 [==============================] - 0s 9ms/step - loss: 0.3759 - accuracy: 0.0071 - val_loss: 0.3377 - val_accuracy: 0.0000e+00
Epoch 9/10
9/9 [==============================] - 0s 9ms/step - loss: 0.3695 - accuracy: 0.0071 - val_loss: 0.3357 - val_accuracy: 0.0000e+00
Epoch 10/10
9/9 [==============================] - 0s 9ms/step - loss: 0.3719 - accuracy: 0.0071 - val_loss: 0.3373 - val_accuracy: 0.0000e+00
```

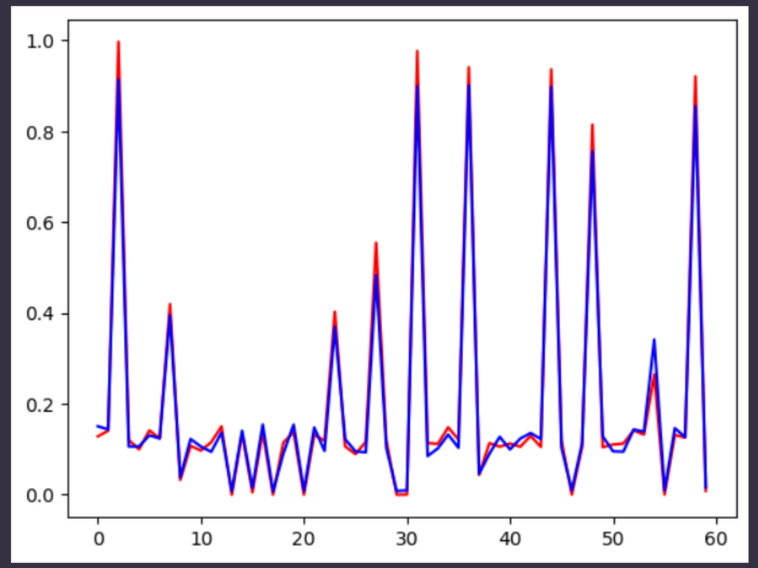


Back to Agenda Page

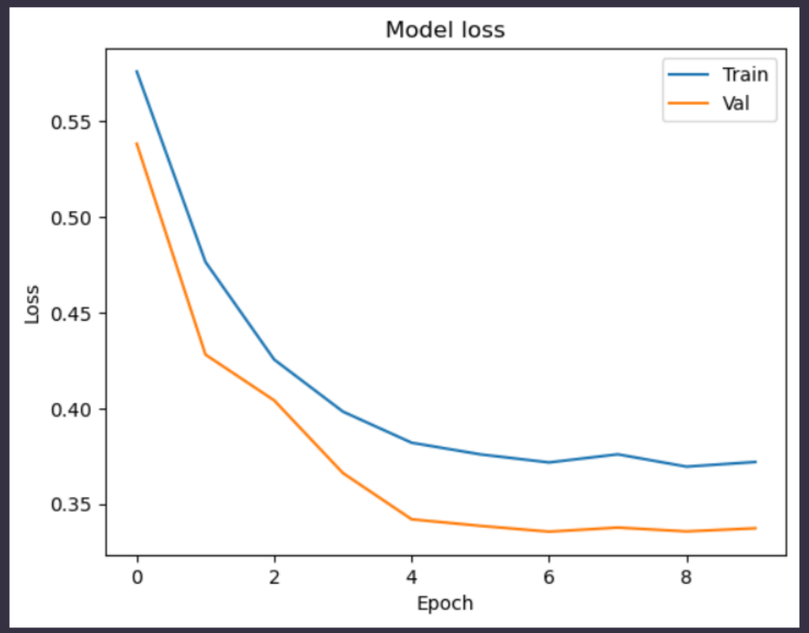
```
test = model_2.predict(X_test)
plt.plot(Y_test, color='r', label="RealData")
plt.plot(test, color='b', label='Predicted Test')
plt.show()
```

[54]

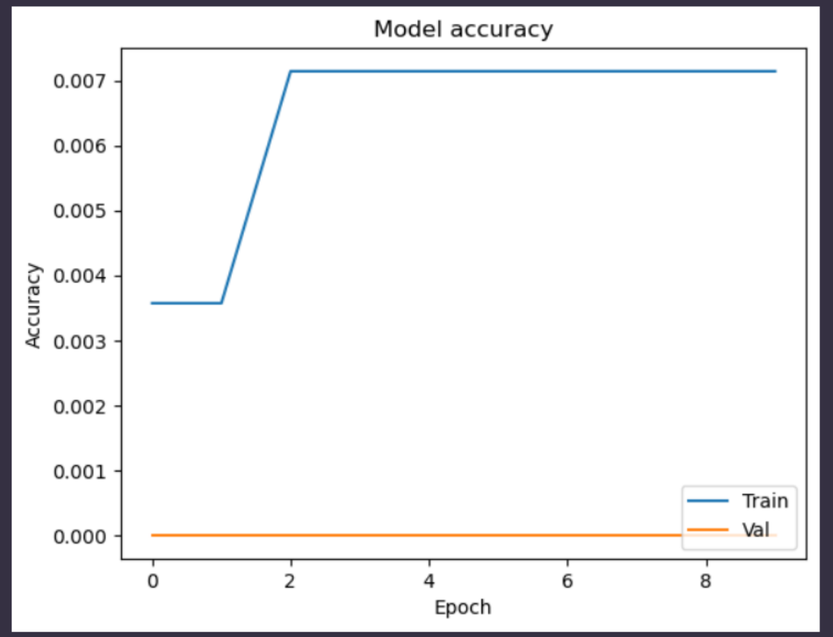
```
2/2 [==============================] - 0s 4ms/step
```




```
plt.plot(hist_2.history['loss'])
plt.plot(hist_2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```

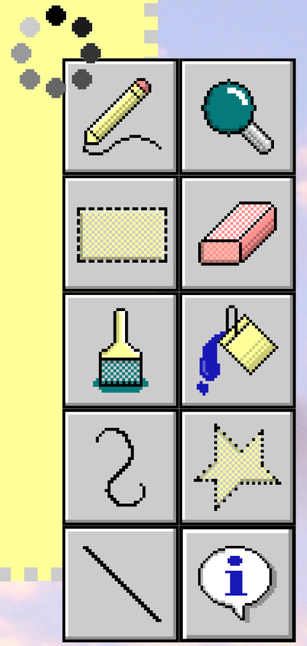



```
plt.plot(hist_2.history['accuracy'])
plt.plot(hist_2.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='lower right')
plt.show()
```

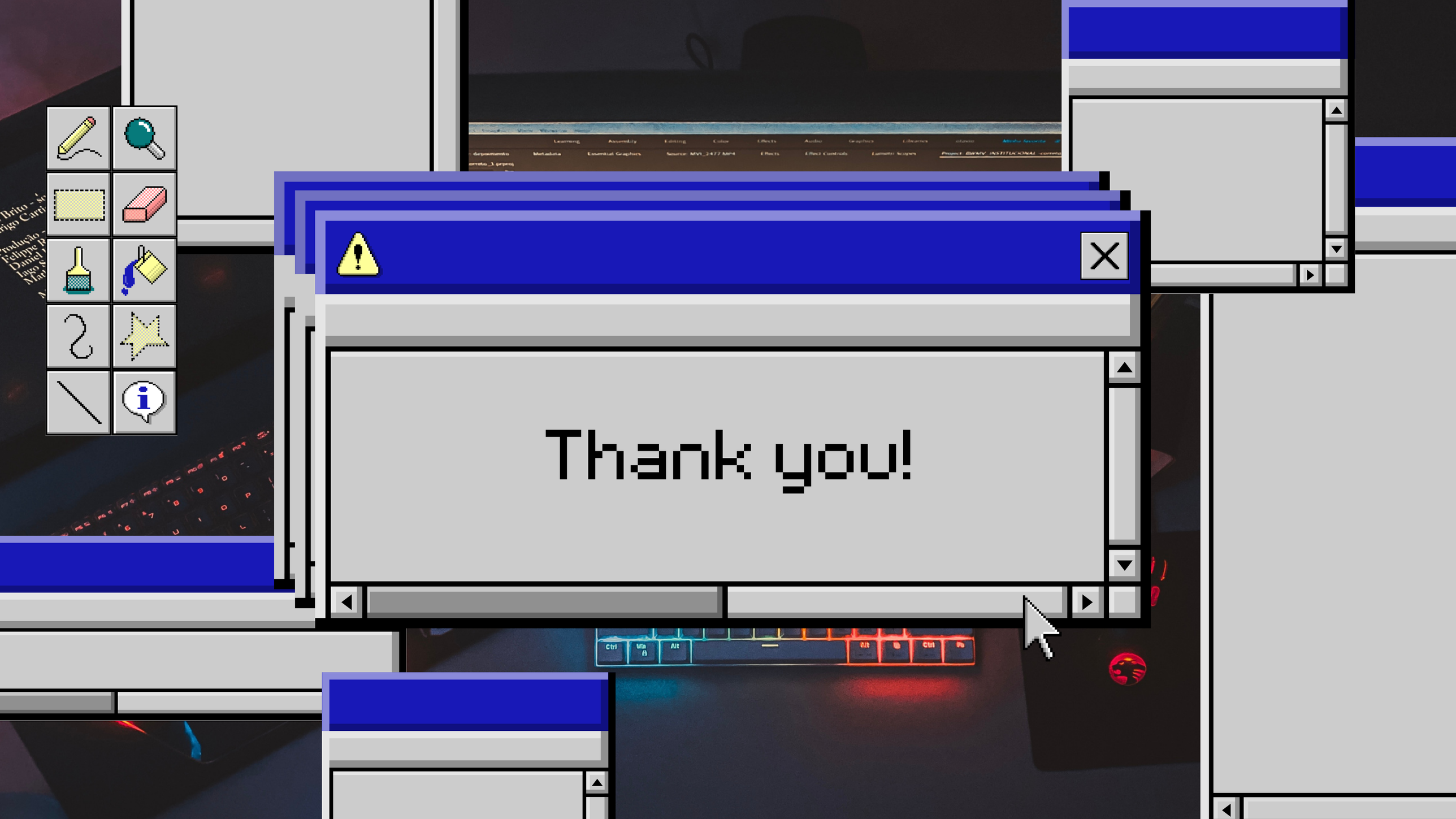


Plot model for X and Y test
Plot model loss &
Plot model accuracy

Conclusion



Back to Agenda Page


Thank you!