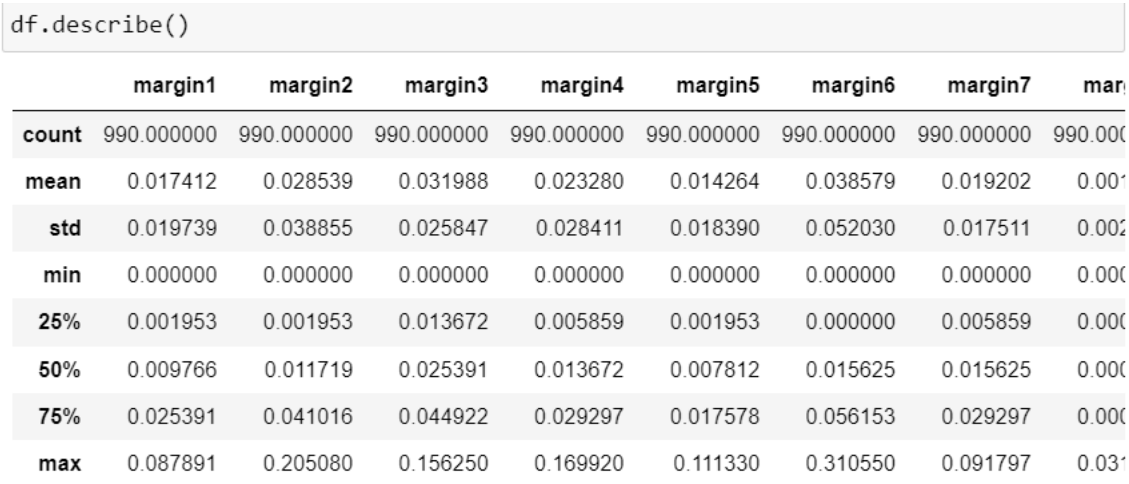
NAME: OMAR SAMEH SAID

ID: 20399132

Project 1 Report

Part I: Data Preparation

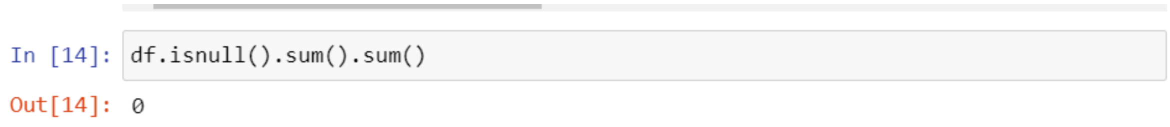
- As we see data is normalized so we don't need to standardize the data



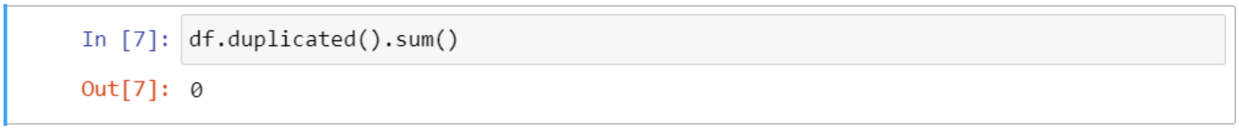
```
df.describe()
```

	margin1	margin2	margin3	margin4	margin5	margin6	margin7	mar
count	990.000000	990.000000	990.000000	990.000000	990.000000	990.000000	990.000000	990.000
mean	0.017412	0.028539	0.031988	0.023280	0.014264	0.038579	0.019202	0.001
std	0.019739	0.038855	0.025847	0.028411	0.018390	0.052030	0.017511	0.002
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	0.001953	0.001953	0.013672	0.005859	0.001953	0.000000	0.005859	0.000
50%	0.009766	0.011719	0.025391	0.013672	0.007812	0.015625	0.015625	0.000
75%	0.025391	0.041016	0.044922	0.029297	0.017578	0.056153	0.029297	0.000
max	0.087891	0.205080	0.156250	0.169920	0.111330	0.310550	0.091797	0.031

- Data is cleaned no missing or duplicates values

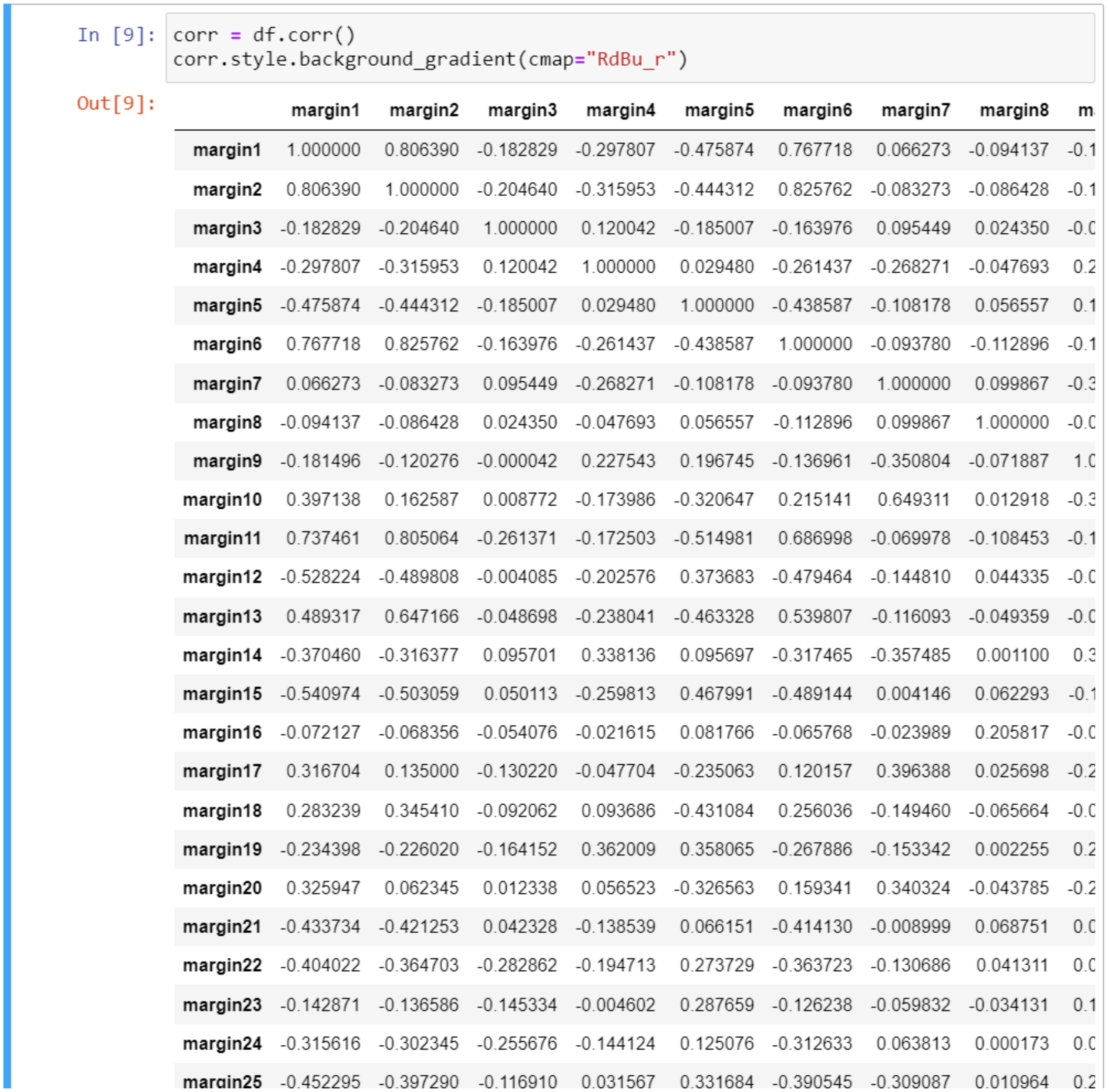


```
In [14]: df.isnull().sum().sum()

Out[14]: 0
```



```
In [7]: df.duplicated().sum()

Out[7]: 0
```

- Sample of correlation analysis

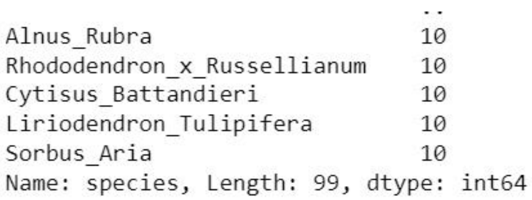


In [9]:
```
corr = df.corr()
corr.style.background_gradient(cmap="RdBu_r")
```

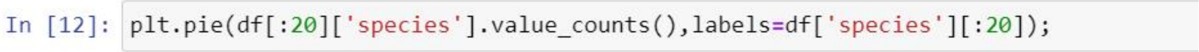
Out[9]:

	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	m
margin1	1.000000	0.806390	-0.182829	-0.297807	-0.475874	0.767718	0.066273	-0.094137	-0.1
margin2	0.806390	1.000000	-0.204640	-0.315953	-0.444312	0.825762	-0.083273	-0.086428	-0.1
margin3	-0.182829	-0.204640	1.000000	0.120042	-0.185007	-0.163976	0.095449	0.024350	-0.0
margin4	-0.297807	-0.315953	0.120042	1.000000	0.029480	-0.261437	-0.268271	-0.047693	0.2
margin5	-0.475874	-0.444312	-0.185007	0.029480	1.000000	-0.438587	-0.108178	0.056557	0.1
margin6	0.767718	0.825762	-0.163976	-0.261437	-0.438587	1.000000	-0.093780	-0.112896	-0.1
margin7	0.066273	-0.083273	0.095449	-0.268271	-0.108178	-0.093780	1.000000	0.099867	-0.3
margin8	-0.094137	-0.086428	0.024350	-0.047693	0.056557	-0.112896	0.099867	1.000000	-0.0
margin9	-0.181496	-0.120276	-0.000042	0.227543	0.196745	-0.136961	-0.350804	-0.071887	1.0
margin10	0.397138	0.162587	0.008772	-0.173986	-0.320647	0.215141	0.649311	0.012918	-0.3
margin11	0.737461	0.805064	-0.261371	-0.172503	-0.514981	0.686998	-0.069978	-0.108453	-0.1
margin12	-0.528224	-0.489808	-0.004085	-0.202576	0.373683	-0.479464	-0.144810	0.044335	-0.0
margin13	0.489317	0.647166	-0.048698	-0.238041	-0.463328	0.539807	-0.116093	-0.049359	-0.0
margin14	-0.370460	-0.316377	0.095701	0.338136	0.095697	-0.317465	-0.357485	0.001100	0.3
margin15	-0.540974	-0.503059	0.050113	-0.259813	0.467991	-0.489144	0.004146	0.062293	-0.1
margin16	-0.072127	-0.068356	-0.054076	-0.021615	0.081766	-0.065768	-0.023989	0.205817	-0.0
margin17	0.316704	0.135000	-0.130220	-0.047704	-0.235063	0.120157	0.396388	0.025698	-0.2
margin18	0.283239	0.345410	-0.092062	0.093686	-0.431084	0.256036	-0.149460	-0.065664	-0.0
margin19	-0.234398	-0.226020	-0.164152	0.362009	0.358065	-0.267886	-0.153342	0.002255	0.2
margin20	0.325947	0.062345	0.012338	0.056523	-0.326563	0.159341	0.340324	-0.043785	-0.2
margin21	-0.433734	-0.421253	0.042328	-0.138539	0.066151	-0.414130	-0.008999	0.068751	0.0
margin22	-0.404022	-0.364703	-0.282862	-0.194713	0.273729	-0.363723	-0.130686	0.041311	0.0
margin23	-0.142871	-0.136586	-0.145334	-0.004602	0.287659	-0.126238	-0.059832	-0.034131	0.1
margin24	-0.315616	-0.302345	-0.255676	-0.144124	0.125076	-0.312633	0.063813	0.000173	0.0
margin25	-0.452295	-0.397290	-0.116910	0.031567	0.331684	-0.390545	-0.309087	0.010964	0.2

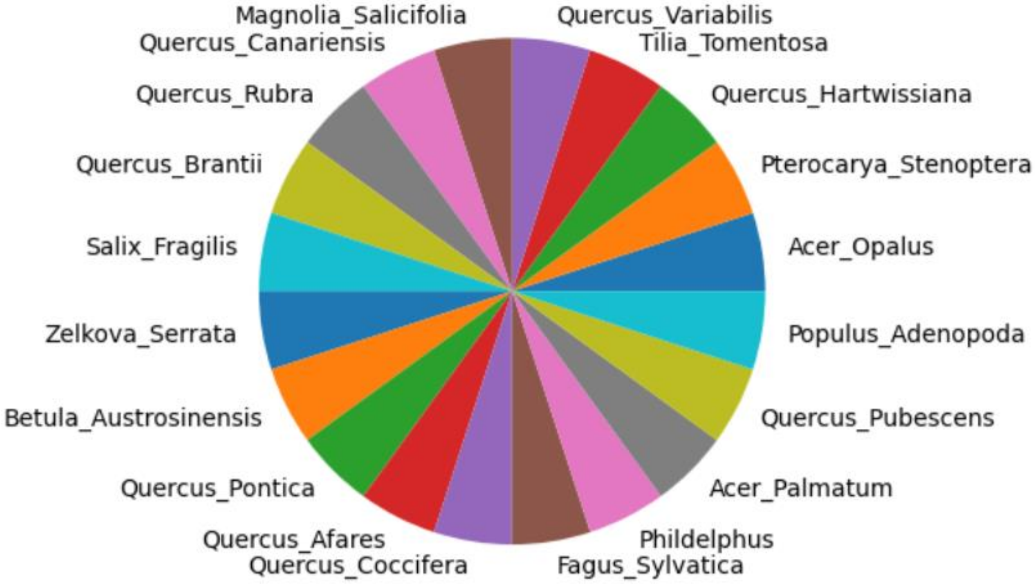
- Visualizing sample of the species we see that the data is balanced



```
                                  ..
Alnus_Rubra                       10
Rhododendron_x_Russellianum       10
Cytisus_Battandieri               10
Liriodendron_Tulipifera           10
Sorbus_Aria                       10
Name: species, Length: 99, dtype: int64
```



```
In [12]: plt.pie(df[:20]['species'].value_counts(),labels=df['species'][:20]);
```



- Displaying some leaf images

Acer_Opalus

Pterocarya_Stenoptera

Quercus_Hartwissiana

Tilia_Tomentosa

Quercus_Variabilis

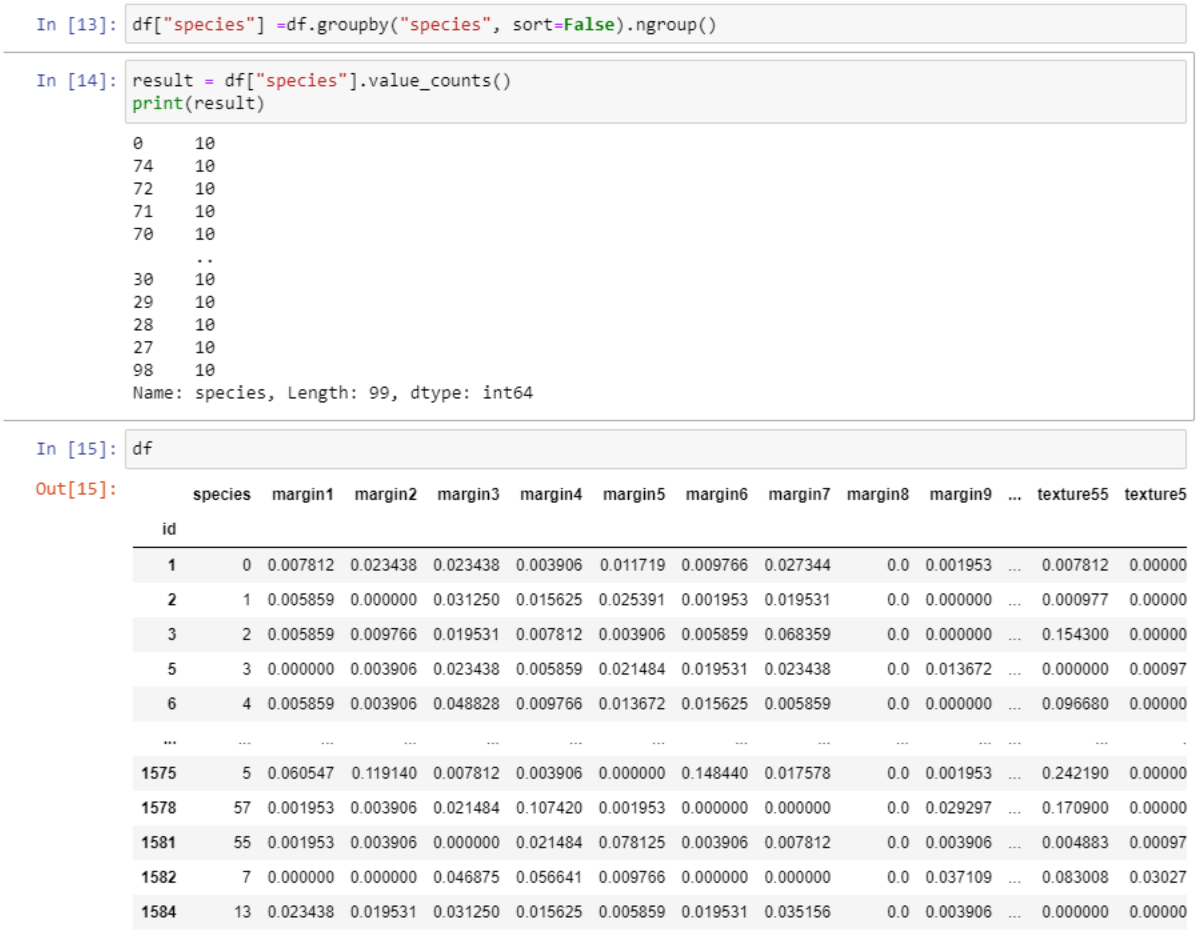
Magnolia_Salicifolia

Quercus_Canariensis

Quercus_Rubra

Quercus_Brantii

- Encode the labels using factorize



```
In [13]: df["species"] =df.groupby("species", sort=False).ngroup()
```

```
In [14]: result = df["species"].value_counts()
         print(result)
```

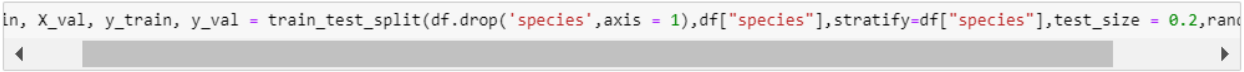
```
0     10
74    10
72    10
71    10
70    10
      ..
30    10
29    10
28    10
27    10
98    10
Name: species, Length: 99, dtype: int64
```

```
In [15]: df
```

Out[15]:

id	species	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	...	texture55	texture5
1	0	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0.0	0.001953	...	0.007812	0.00000
2	1	0.005859	0.000000	0.031250	0.015625	0.025391	0.001953	0.019531	0.0	0.000000	...	0.000977	0.00000
3	2	0.005859	0.009766	0.019531	0.007812	0.003906	0.005859	0.068359	0.0	0.000000	...	0.154300	0.00000
5	3	0.000000	0.003906	0.023438	0.005859	0.021484	0.019531	0.023438	0.0	0.013672	...	0.000000	0.00097
6	4	0.005859	0.003906	0.048828	0.009766	0.013672	0.015625	0.005859	0.0	0.000000	...	0.096680	0.00000
...	...	...	...	...	...	...	...	...	...	...	...	...	.
1575	5	0.060547	0.119140	0.007812	0.003906	0.000000	0.148440	0.017578	0.0	0.001953	...	0.242190	0.00000
1578	57	0.001953	0.003906	0.021484	0.107420	0.001953	0.000000	0.000000	0.0	0.029297	...	0.170900	0.00000
1581	55	0.001953	0.003906	0.000000	0.021484	0.078125	0.003906	0.007812	0.0	0.003906	...	0.004883	0.00097
1582	7	0.000000	0.000000	0.046875	0.056641	0.009766	0.000000	0.000000	0.0	0.037109	...	0.083008	0.03027
1584	13	0.023438	0.019531	0.031250	0.015625	0.005859	0.019531	0.035156	0.0	0.003906	...	0.000000	0.00000

- dividing the data into a training and test set using approximately 80% for training



```
in, X_val, y_train, y_val = train_test_split(df.drop('species',axis = 1),df["species"],stratify=df["species"],test_size = 0.2,ranc
```

Part II: Training a neural network

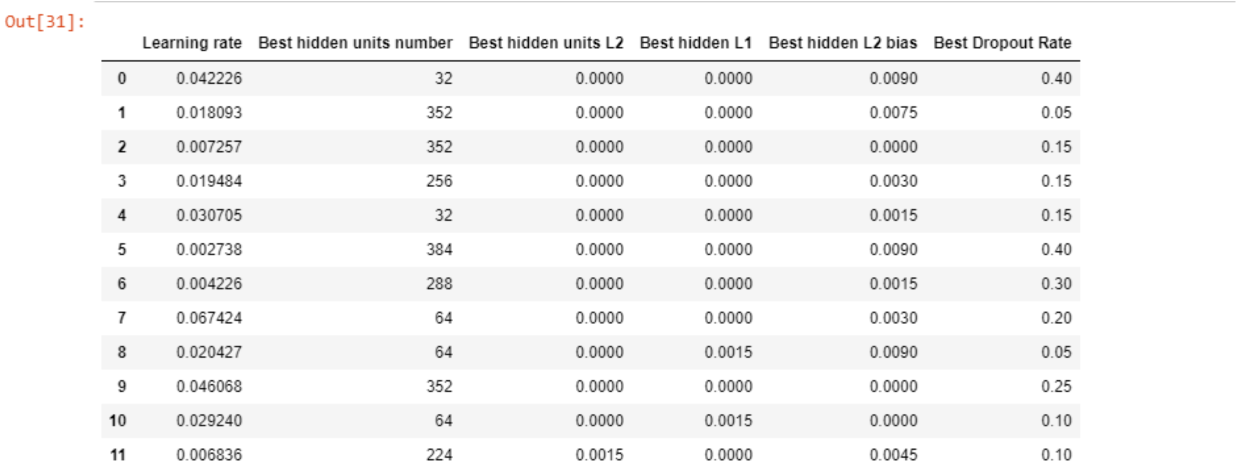
I used Keras tuner to choose the best hyperparameters.

Tuned hyperparameters

- hidden units
- Dropout
- Learning rate
- l1_penalty_hidden
- l2_penalty_hidden
- l2_penalty_hidden_bias

using hyperband which is faster than grid search and better than random search

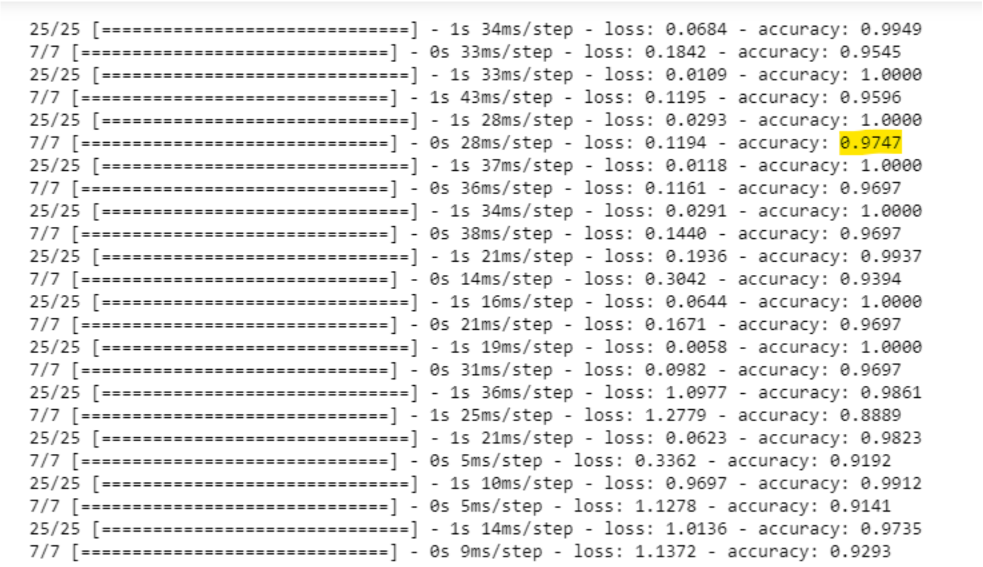
after getting the best hyperparameter combination for 12 model



Out[31]:

	Learning rate	Best hidden units number	Best hidden units L2	Best hidden L1	Best hidden L2 bias	Best Dropout Rate
0	0.042226	32	0.0000	0.0000	0.0090	0.40
1	0.018093	352	0.0000	0.0000	0.0075	0.05
2	0.007257	352	0.0000	0.0000	0.0000	0.15
3	0.019484	256	0.0000	0.0000	0.0030	0.15
4	0.030705	32	0.0000	0.0000	0.0015	0.15
5	0.002738	384	0.0000	0.0000	0.0090	0.40
6	0.004226	288	0.0000	0.0000	0.0015	0.30
7	0.067424	64	0.0000	0.0000	0.0030	0.20
8	0.020427	64	0.0000	0.0015	0.0090	0.05
9	0.046068	352	0.0000	0.0000	0.0000	0.25
10	0.029240	64	0.0000	0.0015	0.0000	0.10
11	0.006836	224	0.0015	0.0000	0.0045	0.10

Then train and evaluate their performance on train/test set



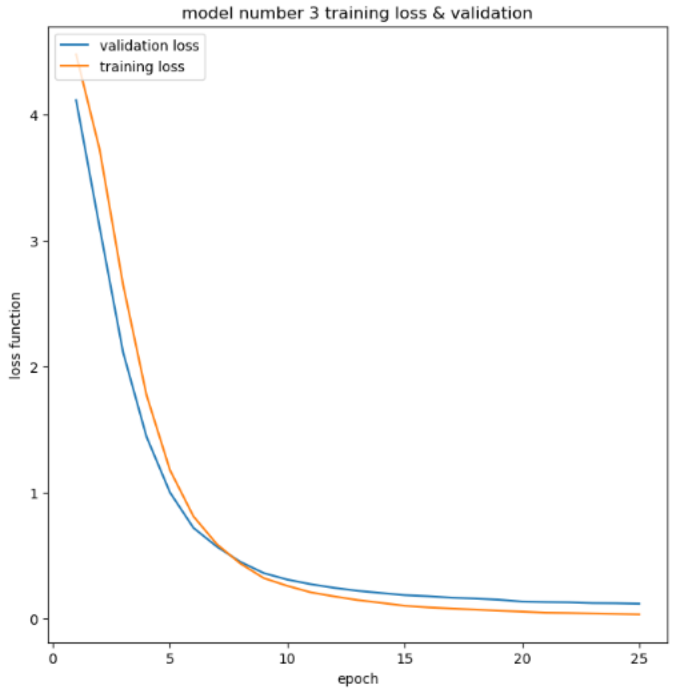
```
25/25 [==============================] - 1s 34ms/step - loss: 0.0684 - accuracy: 0.9949
7/7 [==============================] - 0s 33ms/step - loss: 0.1842 - accuracy: 0.9545
25/25 [==============================] - 1s 33ms/step - loss: 0.0109 - accuracy: 1.0000
7/7 [==============================] - 1s 43ms/step - loss: 0.1195 - accuracy: 0.9596
25/25 [==============================] - 1s 28ms/step - loss: 0.0293 - accuracy: 1.0000
7/7 [==============================] - 0s 28ms/step - loss: 0.1194 - accuracy: 0.9747
25/25 [==============================] - 1s 37ms/step - loss: 0.0118 - accuracy: 1.0000
7/7 [==============================] - 0s 36ms/step - loss: 0.1161 - accuracy: 0.9697
25/25 [==============================] - 1s 34ms/step - loss: 0.0291 - accuracy: 1.0000
7/7 [==============================] - 0s 38ms/step - loss: 0.1440 - accuracy: 0.9697
25/25 [==============================] - 1s 21ms/step - loss: 0.1936 - accuracy: 0.9937
7/7 [==============================] - 0s 14ms/step - loss: 0.3042 - accuracy: 0.9394
25/25 [==============================] - 1s 16ms/step - loss: 0.0644 - accuracy: 1.0000
7/7 [==============================] - 0s 21ms/step - loss: 0.1671 - accuracy: 0.9697
25/25 [==============================] - 1s 19ms/step - loss: 0.0058 - accuracy: 1.0000
7/7 [==============================] - 0s 31ms/step - loss: 0.0982 - accuracy: 0.9697
25/25 [==============================] - 1s 36ms/step - loss: 1.0977 - accuracy: 0.9861
7/7 [==============================] - 1s 25ms/step - loss: 1.2779 - accuracy: 0.8889
25/25 [==============================] - 1s 21ms/step - loss: 0.0623 - accuracy: 0.9823
7/7 [==============================] - 0s 5ms/step - loss: 0.3362 - accuracy: 0.9192
25/25 [==============================] - 1s 10ms/step - loss: 0.9697 - accuracy: 0.9912
7/7 [==============================] - 0s 5ms/step - loss: 1.1278 - accuracy: 0.9141
25/25 [==============================] - 1s 14ms/step - loss: 1.0136 - accuracy: 0.9735
7/7 [==============================] - 0s 9ms/step - loss: 1.1372 - accuracy: 0.9293
```

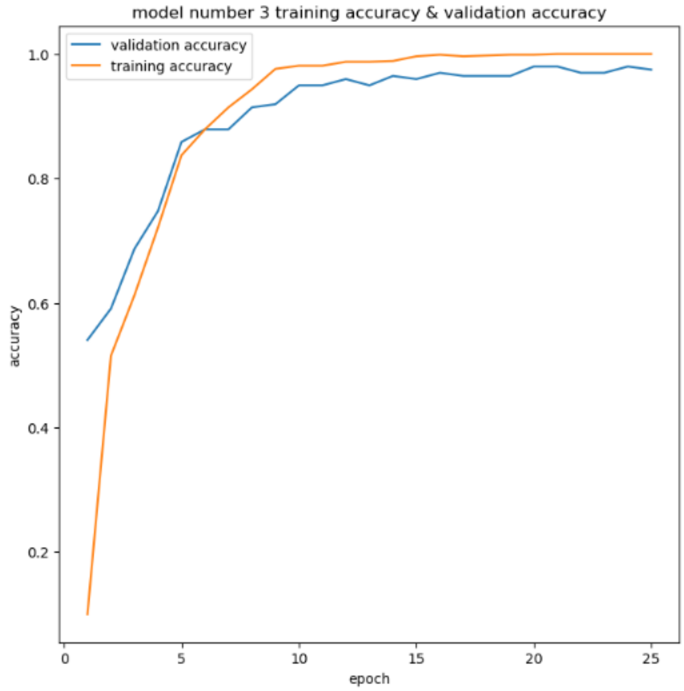
The Model no.3 won with accuracy on test set = 0.9747

The hyper parameter used

Learning rate	hidden units number	L1	L2	L2 bias	Dropout Rate
0.007257	352	0.0000	0.0000	0.0000	0.15

the training curves (accuracy) of this model





The curves of the rest of the models is in the notebook