In [1]:
```
# Import our dependencies
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,OneHotEncoder, MinMaxScaler
import pandas as pd
import tensorflow as tf
import numpy as np

# Import our input dataset
df = pd.read_csv('./pitcher_salaries_cleaned.csv')
df.head()
```

Out[1]:

	Year	Full Name	Age	Salary	ERA	Hits	Earned Runs	Strike Outs	Home Runs	Wins	Losses	Outs Pitched	Batters Faced by Pitcher	Games Finished	Weight	I
0	1990	AbbottJim	23	185000	4.51	246	106	105	16	10	14	635	925	0	200	
1	1990	AbbottPaul	23	100000	5.97	37	23	25	0	0	5	104	162	0	185	
2	1990	AldredScott	22	100000	3.77	13	6	7	0	1	2	43	63	0	195	
3	1990	AndersonAllan	26	300000	4.53	214	95	82	20	7	18	566	797	0	178	
4	1990	AppierKevin	23	100000	2.76	179	57	127	13	12	8	557	784	1	180	

Create Salary Brackets

In [2]:
```
# look at distribution of salaries (suppressing scientific notation)
df['Salary'].describe().apply(lambda x: format(x, 'f'))
```

Out[2]:
```
count        4937.000000
mean      3011304.443387
std       4265619.190449
min        100000.000000
25%        327000.000000
50%        980000.000000
75%       4000000.000000
max      33000000.000000
Name: Salary, dtype: object
```

In [3]:
```
# create salary brackets and labels
bins = [0, 499999, 4999999, 9999999, 34999999]
labels = ['low', 'mid', 'high', 'top']
```

In [4]:
```
# apply salary brackets
df['salBin'] = pd.cut(df['Salary'], bins=bins, labels=labels)
df
```

Out[4]:

	Year	Full Name	Age	Salary	ERA	Hits	Earned Runs	Strike Outs	Home Runs	Wins	Losses	Outs Pitched	Batters Faced by Pitcher	Games Finished
0	1990	AbbottJim	23	185000	4.51	246	106	105	16	10	14	635	925	0
1	1990	AbbottPaul	23	100000	5.97	37	23	25	0	0	5	104	162	0
2	1990	AldredScott	22	100000	3.77	13	6	7	0	1	2	43	63	0
3	1990	AndersonAllan	26	300000	4.53	214	95	82	20	7	18	566	797	0
4	1990	AppierKevin	23	100000	2.76	179	57	127	13	12	8	557	784	1

	Year	Full Name	Age	Salary	ERA	Hits	Earned Runs	Strike Outs	Home Runs	Wins	Losses	Outs Pitched	Batters Faced by Pitcher	Games Finished
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4932	2016	WorleyVance	29	2600000	3.53	84	34	56	11	2	2	260	365	13
4933	2016	WrightMike	26	510500	5.79	81	48	50	12	3	4	224	328	5
4934	2016	WrightSteven	32	514500	3.33	138	58	127	12	13	6	470	656	0
4935	2016	YoungChris	37	4250000	6.19	104	61	94	28	3	9	266	406	7
4936	2016	ZimmermannJordan	30	18000000	4.87	118	57	66	14	9	7	316	450	1

4937 rows × 20 columns

In [5]:
```
### Drop unnecessary columns
df= df.drop(["Full Name","Team","League","Age","Year","Salary"],1)
df.head()
```

```
C:\Users\alyss\anaconda3\envs\mlenv\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: In a future
version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

Out[5]:

	ERA	Hits	Earned Runs	Strike Outs	Home Runs	Wins	Losses	Outs Pitched	Batters Faced by Pitcher	Games Finished	Weight	Height	Games Started	salBin
0	4.51	246	106	105	16	10	14	635	925	0	200	75	33	low
1	5.97	37	23	25	0	0	5	104	162	0	185	75	7	low
2	3.77	13	6	7	0	1	2	43	63	0	195	76	3	low
3	4.53	214	95	82	20	7	18	566	797	0	178	71	31	low
4	2.76	179	57	127	13	12	8	557	784	1	180	74	24	low

Reduce number of rows

kept getting error in one-hot encoding, `ValueError: Buffer has wrong number of dimensions (expected 1, got 2)`

some suggested reducing sample size would solve issue (https://github.com/lmcinnes/umap/issues/496)

-- Update: reducing sample size did not solve issue with one-hot encoding

Encode Salary Bins column

In [6]:
```
# use get_dummies to one-hot encode the salarybin column
encoded_df=pd.get_dummies(df,columns=['salBin'],prefix="salBin")
encoded_df
```

Out[6]:

	ERA	Hits	Earned Runs	Strike Outs	Home Runs	Wins	Losses	Outs Pitched	Batters Faced by Pitcher	Games Finished	Weight	Height	Games Started	salBin_low	salB
0	4.51	246	106	105	16	10	14	635	925	0	200	75	33	1	

	ERA	Hits	Earned Runs	Strike Outs	Home Runs	Wins	Losses	Outs Pitched	Batters Faced by Pitcher	Games Finished	Weight	Height	Games Started	salBin_low	salE
1	5.97	37	23	25	0	0	5	104	162	0	185	75	7	1	
2	3.77	13	6	7	0	1	2	43	63	0	195	76	3	1	
3	4.53	214	95	82	20	7	18	566	797	0	178	71	31	1	
4	2.76	179	57	127	13	12	8	557	784	1	180	74	24	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4932	3.53	84	34	56	11	2	2	260	365	13	240	74	4	0	
4933	5.79	81	48	50	12	3	4	224	328	5	240	78	12	0	
4934	3.33	138	58	127	12	13	6	470	656	0	215	74	24	0	
4935	6.19	104	61	94	28	3	9	266	406	7	255	82	13	0	
4936	4.87	118	57	66	14	9	7	316	450	1	225	74	18	0	

4937 rows × 17 columns

Split Features/Target & Training/Testing Sets

Split into features and target

- **y variable**: Our target variables, `Salary-Bin_low` , `Salary-Bin_mid` , `Salary-Bin_high` , `Salary-Bin_top`
- **X variable**: Our features

In [7]:
```
# Split our preprocessed data into our features and target arrays
y = encoded_df[["salBin_low","salBin_mid","salBin_high","salBin_top"]].values
X = encoded_df.drop(["salBin_low","salBin_mid","salBin_high","salBin_top"],1).values

# Split the preprocessed data into a training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

```
C:\Users\alyss\anaconda3\envs\mlenv\lib\site-packages\ipykernel_launcher.py:3: FutureWarning: In a future
version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  This is separate from the ipykernel package so we can avoid doing imports until
```

Build and Instantiate `StandardScaler` object, then standardize numerical features

In [8]:
```
# Create a StandardScaler instance
scaler = StandardScaler()

# Fit the StandardScaler
X_scaler = scaler.fit(X_train)

# Scale the data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

Build Neural Net Framework

In [9]:
```
# Define the model - deep neural net
```

```
number_input_features = len(X_train[0])
hidden_nodes_layer1 = 144
hidden_nodes_layer2 = 64
hidden_nodes_layer3 = 16


nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation="relu")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="relu"))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation="relu"))

# Output layer
nn.add(tf.keras.layers.Dense(units=4, activation="softmax"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 144)               2016

 dense_1 (Dense)             (None, 64)                9280

 dense_2 (Dense)             (None, 16)                1040

 dense_3 (Dense)             (None, 4)                 68

=================================================================
Total params: 12,404
Trainable params: 12,404
Non-trainable params: 0
_________________________________________________________________
```

Compile the Model

In [10]:

```
# Compile the model
nn.compile(loss="CategoricalCrossentropy", optimizer="adam", metrics=["accuracy"])
```

Train the model

In [11]:

```
# Train the model
fit_model = nn.fit(X_train,y_train,epochs=200)
```

```
Epoch 1/200
116/116 [==============================] - 0s 694us/step - loss: 3.1255 - accuracy: 0.3825
Epoch 2/200
116/116 [==============================] - 0s 652us/step - loss: 1.6097 - accuracy: 0.4211
Epoch 3/200
116/116 [==============================] - 0s 653us/step - loss: 2.0838 - accuracy: 0.4068
Epoch 4/200
116/116 [==============================] - 0s 661us/step - loss: 1.4415 - accuracy: 0.4306
Epoch 5/200
116/116 [==============================] - 0s 678us/step - loss: 1.4461 - accuracy: 0.4246
Epoch 6/200
116/116 [==============================] - 0s 661us/step - loss: 1.3300 - accuracy: 0.4403
Epoch 7/200
116/116 [==============================] - 0s 652us/step - loss: 1.3909 - accuracy: 0.4398
```

```
Epoch 8/200
116/116 [==============================] - 0s 652us/step - loss: 1.4521 - accuracy: 0.4276
Epoch 9/200
116/116 [==============================] - 0s 643us/step - loss: 1.2695 - accuracy: 0.4533
Epoch 10/200
116/116 [==============================] - 0s 652us/step - loss: 1.2636 - accuracy: 0.4449
Epoch 11/200
116/116 [==============================] - 0s 635us/step - loss: 1.2759 - accuracy: 0.4543
Epoch 12/200
116/116 [==============================] - 0s 670us/step - loss: 1.2756 - accuracy: 0.4525
Epoch 13/200
116/116 [==============================] - 0s 652us/step - loss: 1.2714 - accuracy: 0.4516
Epoch 14/200
116/116 [==============================] - 0s 722us/step - loss: 1.2531 - accuracy: 0.4492
Epoch 15/200
116/116 [==============================] - 0s 687us/step - loss: 1.2556 - accuracy: 0.4460
Epoch 16/200
116/116 [==============================] - 0s 670us/step - loss: 1.2367 - accuracy: 0.4500
Epoch 17/200
116/116 [==============================] - 0s 652us/step - loss: 1.2648 - accuracy: 0.4473
Epoch 18/200
116/116 [==============================] - 0s 652us/step - loss: 1.2648 - accuracy: 0.4508
Epoch 19/200
116/116 [==============================] - 0s 652us/step - loss: 1.1827 - accuracy: 0.4662
Epoch 20/200
116/116 [==============================] - 0s 661us/step - loss: 1.1858 - accuracy: 0.4627
Epoch 21/200
116/116 [==============================] - 0s 661us/step - loss: 1.1938 - accuracy: 0.4595
Epoch 22/200
116/116 [==============================] - 0s 670us/step - loss: 1.2506 - accuracy: 0.4446
Epoch 23/200
116/116 [==============================] - 0s 661us/step - loss: 1.2491 - accuracy: 0.4557
Epoch 24/200
116/116 [==============================] - 0s 661us/step - loss: 1.1609 - accuracy: 0.4638
Epoch 25/200
116/116 [==============================] - 0s 661us/step - loss: 1.1736 - accuracy: 0.4557
Epoch 26/200
116/116 [==============================] - 0s 652us/step - loss: 1.2186 - accuracy: 0.4514
Epoch 27/200
116/116 [==============================] - 0s 678us/step - loss: 1.2256 - accuracy: 0.4557
Epoch 28/200
116/116 [==============================] - 0s 670us/step - loss: 1.1917 - accuracy: 0.4471
Epoch 29/200
116/116 [==============================] - 0s 661us/step - loss: 1.1659 - accuracy: 0.4630
Epoch 30/200
116/116 [==============================] - 0s 652us/step - loss: 1.1651 - accuracy: 0.4592
Epoch 31/200
116/116 [==============================] - 0s 678us/step - loss: 1.1517 - accuracy: 0.4603
Epoch 32/200
116/116 [==============================] - 0s 670us/step - loss: 1.1366 - accuracy: 0.4652
Epoch 33/200
116/116 [==============================] - 0s 670us/step - loss: 1.1567 - accuracy: 0.4749
Epoch 34/200
116/116 [==============================] - 0s 704us/step - loss: 1.1572 - accuracy: 0.4627
Epoch 35/200
116/116 [==============================] - 0s 670us/step - loss: 1.1424 - accuracy: 0.4714
Epoch 36/200
116/116 [==============================] - 0s 652us/step - loss: 1.1412 - accuracy: 0.4700
Epoch 37/200
116/116 [==============================] - 0s 670us/step - loss: 1.1277 - accuracy: 0.4787
Epoch 38/200
116/116 [==============================] - 0s 661us/step - loss: 1.1446 - accuracy: 0.4638
Epoch 39/200
116/116 [==============================] - 0s 670us/step - loss: 1.1328 - accuracy: 0.4716
Epoch 40/200
116/116 [==============================] - 0s 678us/step - loss: 1.1331 - accuracy: 0.4679
Epoch 41/200
116/116 [==============================] - 0s 687us/step - loss: 1.1166 - accuracy: 0.4754
Epoch 42/200
116/116 [==============================] - 0s 661us/step - loss: 1.1351 - accuracy: 0.4654
Epoch 43/200
116/116 [==============================] - 0s 652us/step - loss: 1.1263 - accuracy: 0.4762
Epoch 44/200
116/116 [==============================] - 0s 852us/step - loss: 1.1257 - accuracy: 0.4762
Epoch 45/200
116/116 [==============================] - 0s 661us/step - loss: 1.1264 - accuracy: 0.4768
```

```
Epoch 46/200
116/116 [==============================] - 0s 652us/step - loss: 1.1451 - accuracy: 0.4565
Epoch 47/200
116/116 [==============================] - 0s 661us/step - loss: 1.1219 - accuracy: 0.4816
Epoch 48/200
116/116 [==============================] - 0s 670us/step - loss: 1.1126 - accuracy: 0.4854
Epoch 49/200
116/116 [==============================] - 0s 670us/step - loss: 1.1155 - accuracy: 0.4846
Epoch 50/200
116/116 [==============================] - 0s 670us/step - loss: 1.1287 - accuracy: 0.4627
Epoch 51/200
116/116 [==============================] - 0s 661us/step - loss: 1.1159 - accuracy: 0.4792
Epoch 52/200
116/116 [==============================] - 0s 661us/step - loss: 1.1111 - accuracy: 0.4741
Epoch 53/200
116/116 [==============================] - 0s 687us/step - loss: 1.1133 - accuracy: 0.4943
Epoch 54/200
116/116 [==============================] - 0s 713us/step - loss: 1.1178 - accuracy: 0.4830
Epoch 55/200
116/116 [==============================] - 0s 696us/step - loss: 1.1106 - accuracy: 0.4787
Epoch 56/200
116/116 [==============================] - 0s 661us/step - loss: 1.1125 - accuracy: 0.4824
Epoch 57/200
116/116 [==============================] - 0s 661us/step - loss: 1.1122 - accuracy: 0.4835
Epoch 58/200
116/116 [==============================] - 0s 652us/step - loss: 1.1126 - accuracy: 0.4714
Epoch 59/200
116/116 [==============================] - 0s 661us/step - loss: 1.1096 - accuracy: 0.4884
Epoch 60/200
116/116 [==============================] - 0s 678us/step - loss: 1.1191 - accuracy: 0.4776
Epoch 61/200
116/116 [==============================] - 0s 661us/step - loss: 1.1414 - accuracy: 0.4619
Epoch 62/200
116/116 [==============================] - 0s 661us/step - loss: 1.1082 - accuracy: 0.4911
Epoch 63/200
116/116 [==============================] - 0s 670us/step - loss: 1.1225 - accuracy: 0.4835
Epoch 64/200
116/116 [==============================] - 0s 669us/step - loss: 1.1139 - accuracy: 0.4887
Epoch 65/200
116/116 [==============================] - 0s 678us/step - loss: 1.1083 - accuracy: 0.4892
Epoch 66/200
116/116 [==============================] - 0s 652us/step - loss: 1.1219 - accuracy: 0.4884
Epoch 67/200
116/116 [==============================] - 0s 687us/step - loss: 1.1102 - accuracy: 0.4895
Epoch 68/200
116/116 [==============================] - 0s 661us/step - loss: 1.1247 - accuracy: 0.4814
Epoch 69/200
116/116 [==============================] - 0s 678us/step - loss: 1.1081 - accuracy: 0.4873
Epoch 70/200
116/116 [==============================] - 0s 678us/step - loss: 1.1062 - accuracy: 0.4892
Epoch 71/200
116/116 [==============================] - 0s 661us/step - loss: 1.1091 - accuracy: 0.4868
Epoch 72/200
116/116 [==============================] - 0s 661us/step - loss: 1.1130 - accuracy: 0.4824
Epoch 73/200
116/116 [==============================] - 0s 661us/step - loss: 1.1037 - accuracy: 0.4860
Epoch 74/200
116/116 [==============================] - 0s 661us/step - loss: 1.1089 - accuracy: 0.4860
Epoch 75/200
116/116 [==============================] - 0s 670us/step - loss: 1.1082 - accuracy: 0.4833
Epoch 76/200
116/116 [==============================] - 0s 670us/step - loss: 1.1151 - accuracy: 0.4846
Epoch 77/200
116/116 [==============================] - 0s 687us/step - loss: 1.1044 - accuracy: 0.4914
Epoch 78/200
116/116 [==============================] - 0s 687us/step - loss: 1.1130 - accuracy: 0.4846
Epoch 79/200
116/116 [==============================] - 0s 687us/step - loss: 1.1096 - accuracy: 0.4878
Epoch 80/200
116/116 [==============================] - 0s 670us/step - loss: 1.1024 - accuracy: 0.4916
Epoch 81/200
116/116 [==============================] - 0s 661us/step - loss: 1.1029 - accuracy: 0.4830
Epoch 82/200
116/116 [==============================] - 0s 661us/step - loss: 1.1045 - accuracy: 0.4914
Epoch 83/200
116/116 [==============================] - 0s 661us/step - loss: 1.1111 - accuracy: 0.4841
```

```
Epoch 84/200
116/116 [==============================] - 0s 661us/step - loss: 1.1086 - accuracy: 0.4905
Epoch 85/200
116/116 [==============================] - 0s 661us/step - loss: 1.1033 - accuracy: 0.4881
Epoch 86/200
116/116 [==============================] - 0s 652us/step - loss: 1.1067 - accuracy: 0.4868
Epoch 87/200
116/116 [==============================] - 0s 678us/step - loss: 1.1024 - accuracy: 0.4927
Epoch 88/200
116/116 [==============================] - 0s 687us/step - loss: 1.1001 - accuracy: 0.4870
Epoch 89/200
116/116 [==============================] - 0s 696us/step - loss: 1.1065 - accuracy: 0.4916
Epoch 90/200
116/116 [==============================] - 0s 913us/step - loss: 1.1014 - accuracy: 0.4830
Epoch 91/200
116/116 [==============================] - 0s 722us/step - loss: 1.1037 - accuracy: 0.4843
Epoch 92/200
116/116 [==============================] - 0s 748us/step - loss: 1.1076 - accuracy: 0.4889
Epoch 93/200
116/116 [==============================] - 0s 765us/step - loss: 1.0978 - accuracy: 0.4892
Epoch 94/200
116/116 [==============================] - 0s 843us/step - loss: 1.1041 - accuracy: 0.4927
Epoch 95/200
116/116 [==============================] - 0s 722us/step - loss: 1.0947 - accuracy: 0.4876
Epoch 96/200
116/116 [==============================] - 0s 696us/step - loss: 1.1024 - accuracy: 0.4835
Epoch 97/200
116/116 [==============================] - 0s 687us/step - loss: 1.0967 - accuracy: 0.4911
Epoch 98/200
116/116 [==============================] - 0s 730us/step - loss: 1.0982 - accuracy: 0.4932
Epoch 99/200
116/116 [==============================] - 0s 678us/step - loss: 1.0985 - accuracy: 0.4895
Epoch 100/200
116/116 [==============================] - 0s 678us/step - loss: 1.1027 - accuracy: 0.4860
Epoch 101/200
116/116 [==============================] - 0s 670us/step - loss: 1.0975 - accuracy: 0.4884
Epoch 102/200
116/116 [==============================] - 0s 678us/step - loss: 1.0927 - accuracy: 0.4876
Epoch 103/200
116/116 [==============================] - 0s 670us/step - loss: 1.0996 - accuracy: 0.4903
Epoch 104/200
116/116 [==============================] - 0s 670us/step - loss: 1.0974 - accuracy: 0.4862
Epoch 105/200
116/116 [==============================] - 0s 661us/step - loss: 1.1026 - accuracy: 0.4876
Epoch 106/200
116/116 [==============================] - 0s 678us/step - loss: 1.0977 - accuracy: 0.4865
Epoch 107/200
116/116 [==============================] - 0s 670us/step - loss: 1.0935 - accuracy: 0.4938
Epoch 108/200
116/116 [==============================] - 0s 670us/step - loss: 1.0989 - accuracy: 0.4816
Epoch 109/200
116/116 [==============================] - 0s 687us/step - loss: 1.0951 - accuracy: 0.4943
Epoch 110/200
116/116 [==============================] - 0s 661us/step - loss: 1.0976 - accuracy: 0.4897
Epoch 111/200
116/116 [==============================] - 0s 678us/step - loss: 1.0901 - accuracy: 0.4884
Epoch 112/200
116/116 [==============================] - 0s 670us/step - loss: 1.0903 - accuracy: 0.4892
Epoch 113/200
116/116 [==============================] - 0s 678us/step - loss: 1.0973 - accuracy: 0.4833
Epoch 114/200
116/116 [==============================] - 0s 670us/step - loss: 1.1010 - accuracy: 0.4922
Epoch 115/200
116/116 [==============================] - 0s 652us/step - loss: 1.0958 - accuracy: 0.4911
Epoch 116/200
116/116 [==============================] - 0s 687us/step - loss: 1.0974 - accuracy: 0.4914
Epoch 117/200
116/116 [==============================] - 0s 696us/step - loss: 1.0917 - accuracy: 0.4922
Epoch 118/200
116/116 [==============================] - 0s 661us/step - loss: 1.0936 - accuracy: 0.4989
Epoch 119/200
116/116 [==============================] - 0s 652us/step - loss: 1.0963 - accuracy: 0.4900
Epoch 120/200
116/116 [==============================] - 0s 678us/step - loss: 1.0928 - accuracy: 0.4900
Epoch 121/200
116/116 [==============================] - 0s 687us/step - loss: 1.0904 - accuracy: 0.5000
```

```
Epoch 122/200
116/116 [==============================] - 0s 678us/step - loss: 1.0905 - accuracy: 0.4819
Epoch 123/200
116/116 [==============================] - 0s 678us/step - loss: 1.0902 - accuracy: 0.4857
Epoch 124/200
116/116 [==============================] - 0s 670us/step - loss: 1.0935 - accuracy: 0.4849
Epoch 125/200
116/116 [==============================] - 0s 696us/step - loss: 1.0981 - accuracy: 0.4889
Epoch 126/200
116/116 [==============================] - 0s 687us/step - loss: 1.0897 - accuracy: 0.4930
Epoch 127/200
116/116 [==============================] - 0s 661us/step - loss: 1.0862 - accuracy: 0.4860
Epoch 128/200
116/116 [==============================] - 0s 861us/step - loss: 1.0897 - accuracy: 0.4954
Epoch 129/200
116/116 [==============================] - 0s 687us/step - loss: 1.0864 - accuracy: 0.4930
Epoch 130/200
116/116 [==============================] - 0s 670us/step - loss: 1.0914 - accuracy: 0.4949
Epoch 131/200
116/116 [==============================] - 0s 670us/step - loss: 1.0919 - accuracy: 0.4854
Epoch 132/200
116/116 [==============================] - 0s 661us/step - loss: 1.0854 - accuracy: 0.4943
Epoch 133/200
116/116 [==============================] - 0s 670us/step - loss: 1.0909 - accuracy: 0.4932
Epoch 134/200
116/116 [==============================] - 0s 670us/step - loss: 1.0824 - accuracy: 0.4981
Epoch 135/200
116/116 [==============================] - 0s 678us/step - loss: 1.0926 - accuracy: 0.4922
Epoch 136/200
116/116 [==============================] - 0s 670us/step - loss: 1.0842 - accuracy: 0.4976
Epoch 137/200
116/116 [==============================] - 0s 661us/step - loss: 1.0864 - accuracy: 0.4932
Epoch 138/200
116/116 [==============================] - 0s 661us/step - loss: 1.0921 - accuracy: 0.4932
Epoch 139/200
116/116 [==============================] - 0s 678us/step - loss: 1.0845 - accuracy: 0.4970
Epoch 140/200
116/116 [==============================] - 0s 661us/step - loss: 1.0915 - accuracy: 0.4951
Epoch 141/200
116/116 [==============================] - 0s 687us/step - loss: 1.0823 - accuracy: 0.4997
Epoch 142/200
116/116 [==============================] - 0s 670us/step - loss: 1.0866 - accuracy: 0.4959
Epoch 143/200
116/116 [==============================] - 0s 678us/step - loss: 1.0944 - accuracy: 0.4954
Epoch 144/200
116/116 [==============================] - 0s 670us/step - loss: 1.0856 - accuracy: 0.4986
Epoch 145/200
116/116 [==============================] - 0s 661us/step - loss: 1.0836 - accuracy: 0.4938
Epoch 146/200
116/116 [==============================] - 0s 661us/step - loss: 1.0848 - accuracy: 0.5032
Epoch 147/200
116/116 [==============================] - 0s 678us/step - loss: 1.0893 - accuracy: 0.4943
Epoch 148/200
116/116 [==============================] - 0s 696us/step - loss: 1.0893 - accuracy: 0.4965
Epoch 149/200
116/116 [==============================] - 0s 730us/step - loss: 1.0798 - accuracy: 0.4984
Epoch 150/200
116/116 [==============================] - 0s 739us/step - loss: 1.0808 - accuracy: 0.4916
Epoch 151/200
116/116 [==============================] - 0s 713us/step - loss: 1.0812 - accuracy: 0.4916
Epoch 152/200
116/116 [==============================] - 0s 687us/step - loss: 1.0812 - accuracy: 0.4892
Epoch 153/200
116/116 [==============================] - 0s 704us/step - loss: 1.0835 - accuracy: 0.5008
Epoch 154/200
116/116 [==============================] - 0s 730us/step - loss: 1.0785 - accuracy: 0.4938
Epoch 155/200
116/116 [==============================] - 0s 809us/step - loss: 1.0746 - accuracy: 0.4908
Epoch 156/200
116/116 [==============================] - 0s 687us/step - loss: 1.0848 - accuracy: 0.4932
Epoch 157/200
116/116 [==============================] - 0s 670us/step - loss: 1.0837 - accuracy: 0.4865
Epoch 158/200
116/116 [==============================] - 0s 704us/step - loss: 1.0765 - accuracy: 0.5005
Epoch 159/200
116/116 [==============================] - 0s 704us/step - loss: 1.0784 - accuracy: 0.5054
```

```
Epoch 160/200
116/116 [==============================] - 0s 696us/step - loss: 1.0755 - accuracy: 0.4954
Epoch 161/200
116/116 [==============================] - 0s 687us/step - loss: 1.0804 - accuracy: 0.4927
Epoch 162/200
116/116 [==============================] - 0s 678us/step - loss: 1.0774 - accuracy: 0.4976
Epoch 163/200
116/116 [==============================] - 0s 722us/step - loss: 1.0747 - accuracy: 0.4927
Epoch 164/200
116/116 [==============================] - 0s 739us/step - loss: 1.0784 - accuracy: 0.4957
Epoch 165/200
116/116 [==============================] - 0s 694us/step - loss: 1.0837 - accuracy: 0.4927
Epoch 166/200
116/116 [==============================] - 0s 661us/step - loss: 1.0718 - accuracy: 0.5008
Epoch 167/200
116/116 [==============================] - 0s 687us/step - loss: 1.0777 - accuracy: 0.4992
Epoch 168/200
116/116 [==============================] - 0s 722us/step - loss: 1.0761 - accuracy: 0.4984
Epoch 169/200
116/116 [==============================] - 0s 730us/step - loss: 1.0739 - accuracy: 0.4997
Epoch 170/200
116/116 [==============================] - 0s 696us/step - loss: 1.0690 - accuracy: 0.5043
Epoch 171/200
116/116 [==============================] - 0s 670us/step - loss: 1.0749 - accuracy: 0.5014
Epoch 172/200
116/116 [==============================] - 0s 835us/step - loss: 1.0805 - accuracy: 0.5005
Epoch 173/200
116/116 [==============================] - 0s 739us/step - loss: 1.0755 - accuracy: 0.4981
Epoch 174/200
116/116 [==============================] - 0s 730us/step - loss: 1.0702 - accuracy: 0.5005
Epoch 175/200
116/116 [==============================] - 0s 704us/step - loss: 1.0713 - accuracy: 0.5014
Epoch 176/200
116/116 [==============================] - 0s 678us/step - loss: 1.0796 - accuracy: 0.4951
Epoch 177/200
116/116 [==============================] - 0s 668us/step - loss: 1.0715 - accuracy: 0.5016
Epoch 178/200
116/116 [==============================] - 0s 872us/step - loss: 1.0747 - accuracy: 0.4957
Epoch 179/200
116/116 [==============================] - 0s 722us/step - loss: 1.0673 - accuracy: 0.5035
Epoch 180/200
116/116 [==============================] - 0s 687us/step - loss: 1.0670 - accuracy: 0.4984
Epoch 181/200
116/116 [==============================] - 0s 661us/step - loss: 1.0677 - accuracy: 0.5046
Epoch 182/200
116/116 [==============================] - 0s 704us/step - loss: 1.0768 - accuracy: 0.4941
Epoch 183/200
116/116 [==============================] - 0s 730us/step - loss: 1.0669 - accuracy: 0.5019
Epoch 184/200
116/116 [==============================] - 0s 730us/step - loss: 1.0676 - accuracy: 0.5027
Epoch 185/200
116/116 [==============================] - 0s 687us/step - loss: 1.0705 - accuracy: 0.4951
Epoch 186/200
116/116 [==============================] - 0s 687us/step - loss: 1.0703 - accuracy: 0.5068
Epoch 187/200
116/116 [==============================] - 0s 652us/step - loss: 1.0737 - accuracy: 0.5049
Epoch 188/200
116/116 [==============================] - 0s 670us/step - loss: 1.0661 - accuracy: 0.5022
Epoch 189/200
116/116 [==============================] - 0s 661us/step - loss: 1.0658 - accuracy: 0.5014
Epoch 190/200
116/116 [==============================] - 0s 687us/step - loss: 1.0673 - accuracy: 0.5030
Epoch 191/200
116/116 [==============================] - 0s 704us/step - loss: 1.0694 - accuracy: 0.5014
Epoch 192/200
116/116 [==============================] - 0s 670us/step - loss: 1.0740 - accuracy: 0.5035
Epoch 193/200
116/116 [==============================] - 0s 670us/step - loss: 1.0712 - accuracy: 0.5078
Epoch 194/200
116/116 [==============================] - 0s 652us/step - loss: 1.0637 - accuracy: 0.5019
Epoch 195/200
116/116 [==============================] - 0s 670us/step - loss: 1.0667 - accuracy: 0.5049
Epoch 196/200
116/116 [==============================] - 0s 695us/step - loss: 1.0643 - accuracy: 0.5095
Epoch 197/200
116/116 [==============================] - 0s 734us/step - loss: 1.0622 - accuracy: 0.5051
```

```
Epoch 198/200
116/116 [==============================] - 0s 696us/step - loss: 1.0628 - accuracy: 0.4954
Epoch 199/200
116/116 [==============================] - 0s 670us/step - loss: 1.0640 - accuracy: 0.5043
Epoch 200/200
116/116 [==============================] - 0s 661us/step - loss: 1.0635 - accuracy: 0.5030
```

In [12]:

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss*100:.2f}%, Accuracy: {model_accuracy*100:.2f}%")
```

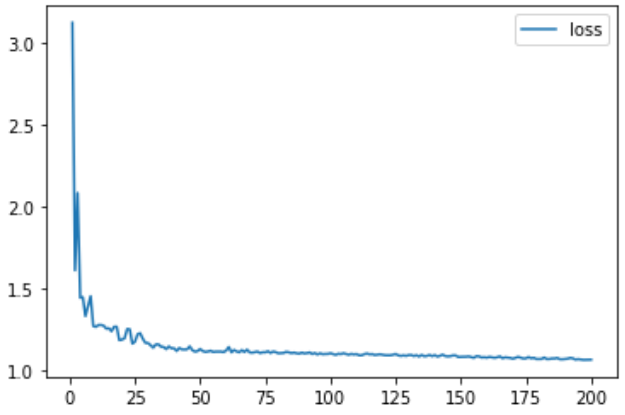
```
39/39 - 0s - loss: 1.2123 - accuracy: 0.4834 - 107ms/epoch - 3ms/step
Loss: 121.23%, Accuracy: 48.34%
```

In [13]:

```
# Create a DataFrame containing training history
history_df = pd.DataFrame(fit_model.history, index=range(1,len(fit_model.history["loss"])+1))

# Plot the loss
history_df.plot(y="loss")
```

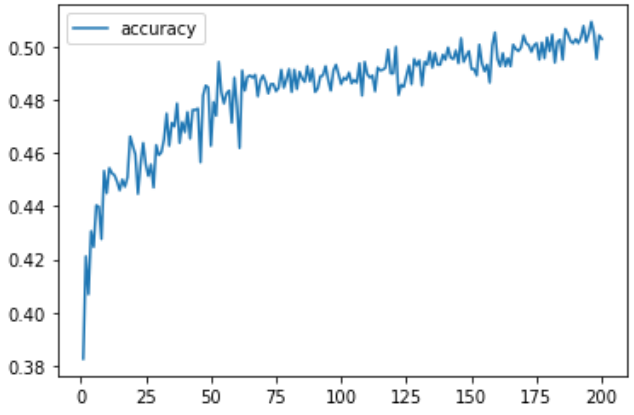
Out[13]: <AxesSubplot:>



In [14]:

```
# Plot the accuracy
history_df.plot(y="accuracy")
```

Out[14]: <AxesSubplot:>



In [ ]: