In [23]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split


from sklearn .linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
```

In [24]:
```
data=pd.read_csv('csgo_round_snapshots.csv')
```

In [25]:
```
data
```

Out[25]:

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	t_grenade_flashbang	ct_grenade_smokegrenade	t_grenade_smokegrenade	ct_grenade_incendiarygrenade	t_grenade_incendiarygrenade	ct_grenade_molotovgrenade	t_grenade_molotovgrenade	ct_grenade_decoygrenade	t_grenade_decoygrenade	round_winner
0	175.00	0.0	0.0	de_dust2	False	500.0	500.0	0.0	0.0	4000.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
1	156.03	0.0	0.0	de_dust2	False	500.0	500.0	400.0	300.0	600.0	...	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
2	96.03	0.0	0.0	de_dust2	False	391.0	400.0	294.0	200.0	750.0	...	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
3	76.03	0.0	0.0	de_dust2	False	391.0	400.0	294.0	200.0	750.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
4	174.97	1.0	0.0	de_dust2	False	500.0	500.0	192.0	0.0	18350.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
122405	15.41	11.0	14.0	de_train	True	200.0	242.0	195.0	359.0	100.0	...	2.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	T
122406	174.93	11.0	15.0	de_train	False	500.0	500.0	95.0	175.0	11500.0	...	2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	T
122407	114.93	11.0	15.0	de_train	False	500.0	500.0	495.0	475.0	1200.0	...	4.0	3.0	5.0	1.0	0.0	0.0	5.0	0.0	0.0	T
122408	94.93	11.0	15.0	de_train	False	500.0	500.0	495.0	475.0	1200.0	...	5.0	0.0	3.0	0.0	0.0	0.0	4.0	0.0	0.0	T
122409	74.93	11.0	15.0	de_train	False	375.0	479.0	395.0	466.0	1100.0	...	3.0	0.0	2.0	0.0	0.0	0.0	3.0	0.0	0.0	T

122410 rows × 97 columns

In [26]:
```
np.sum(np.sum(data.isnull()))
```

Out[26]:
```
0
```

In [27]:
```
#data.drop(data.index[122410].axis=0,inplace=True)
```

In [28]:
```
data.drop(data.select_dtypes(np.number),axis=1)
```

Out[28]:

	map	bomb_planted	round_winner
0	de_dust2	False	CT
1	de_dust2	False	CT
2	de_dust2	False	CT
3	de_dust2	False	CT
4	de_dust2	False	CT
...	...	...	...
122405	de_train	True	T
122406	de_train	False	T
122407	de_train	False	T
122408	de_train	False	T
122409	de_train	False	T

122410 rows × 3 columns

In [29]:
```
data['bomb_planted']=data['bomb_planted'].astype(np.int16)
```

In [30]:
```
data
```

Out[30]:

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	t_grenade_flashbang	ct_grenade_smokegrenade	t_grenade_smokegrenade	ct_grenade_incendiarygrenade	t_grenade_incendiarygrenade	ct_grenade_molotovgrenade	t_grenade_molotovgrenade	ct_grenade_decoygrenade	t_grenade_decoygrenade	round_winner
0	175.00	0.0	0.0	de_dust2	0	500.0	500.0	0.0	0.0	4000.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
1	156.03	0.0	0.0	de_dust2	0	500.0	500.0	400.0	300.0	600.0	...	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
2	96.03	0.0	0.0	de_dust2	0	391.0	400.0	294.0	200.0	750.0	...	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
3	76.03	0.0	0.0	de_dust2	0	391.0	400.0	294.0	200.0	750.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
4	174.97	1.0	0.0	de_dust2	0	500.0	500.0	192.0	0.0	18350.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	CT
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
122405	15.41	11.0	14.0	de_train	1	200.0	242.0	195.0	359.0	100.0	...	2.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	T
122406	174.93	11.0	15.0	de_train	0	500.0	500.0	95.0	175.0	11500.0	...	2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	T
122407	114.93	11.0	15.0	de_train	0	500.0	500.0	495.0	475.0	1200.0	...	4.0	3.0	5.0	1.0	0.0	0.0	5.0	0.0	0.0	T
122408	94.93	11.0	15.0	de_train	0	500.0	500.0	495.0	475.0	1200.0	...	5.0	0.0	3.0	0.0	0.0	0.0	4.0	0.0	0.0	T
122409	74.93	11.0	15.0	de_train	0	375.0	479.0	395.0	466.0	1100.0	...	3.0	0.0	2.0	0.0	0.0	0.0	3.0	0.0	0.0	T

122410 rows × 97 columns

In [31]:
```
encoder=LabelEncoder()

data['map']=encoder.fit_transform(data['map'])
map_mappings={index: label for index, label in enumerate(encoder.classes_)}

map_mappings
```

Out[31]:
```
{0: 'de_cache',
 1: 'de_dust2',
 2: 'de_inferno',
 3: 'de_mirage',
 4: 'de_nuke',
 5: 'de_overpass',
 6: 'de_train',
 7: 'de_vertigo'}
```

In [32]:
```
data['round_winner']=encoder.fit_transform(data['round_winner'])
round_winner_mappings={index: label for index, label in enumerate(encoder.classes_)}

round_winner_mappings
```

Out[32]:
```
{0: 'CT', 1: 'T'}
```

In [33]:
```
y=data['round_winner']
X=data.drop('round_winner',axis=1)
```

In [34]:
```
scaler=RobustScaler()
X=scaler.fit_transform(X)
pd.DataFrame(X)
```

Out[34]:

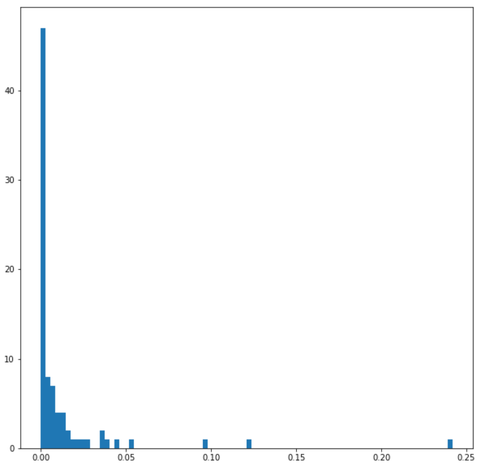
	0	1	2	3	4	5	6	7	8	9	...	86	87	88	89	90	91	92	93	94	95
0	0.715105	-0.857143	-0.857143	-0.666667	0.0	0.000000	0.000000	-1.291096	-1.136054	-0.112782	...	-0.333333	-0.333333	-0.333333	-0.333333	0.0	0.0	0.0	-0.5	0.0	0.0
1	0.545726	-0.857143	-0.857143	-0.666667	0.0	0.000000	0.000000	0.078767	-0.115646	-0.368421	...	-0.333333	-0.333333	-0.333333	0.333333	0.0	0.0	0.0	-0.5	0.0	0.0
2	0.010000	-0.857143	-0.857143	-0.666667	0.0	-0.726667	-0.561798	-0.284247	-0.455782	-0.357143	...	-0.333333	-0.333333	-0.333333	0.333333	0.0	0.0	0.0	-0.5	0.0	0.0
3	-0.168575	-0.857143	-0.857143	-0.666667	0.0	-0.726667	-0.561798	-0.284247	-0.455782	-0.357143	...	-0.333333	-0.333333	-0.333333	-0.333333	0.0	0.0	0.0	-0.5	0.0	0.0
4	0.714837	-0.714286	-0.857143	-0.666667	0.0	0.000000	0.000000	-0.633562	-1.136054	0.966165	...	-0.333333	-0.333333	-0.333333	-0.333333	0.0	0.0	0.0	-0.5	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
122405	-0.709837	0.714286	1.142857	1.000000	1.0	-2.000000	-1.449438	-0.623288	0.085034	-0.406015	...	0.000000	0.333333	0.000000	0.000000	0.0	0.0	0.0	-0.5	0.0	0.0
122406	0.714480	0.714286	1.285714	1.000000	0.0	0.000000	0.000000	-0.965753	-0.540816	0.451128	...	0.000000	0.333333	0.000000	-0.333333	0.0	0.0	0.0	-0.5	0.0	0.0
122407	0.178754	0.714286	1.285714	1.000000	0.0	0.000000	0.000000	0.404110	0.479592	-0.323308	...	1.000000	1.000000	0.666667	1.333333	0.5	0.0	0.0	2.0	0.0	0.0
122408	0.000179	0.714286	1.285714	1.000000	0.0	0.000000	0.000000	0.404110	0.479592	-0.323308	...	0.000000	1.333333	-0.333333	0.666667	0.0	0.0	0.0	1.5	0.0	0.0
122409	-0.178397	0.714286	1.285714	1.000000	0.0	-0.833333	-0.117978	0.061644	0.448980	-0.330827	...	-0.333333	0.666667	-0.333333	0.333333	0.0	0.0	0.0	1.0	0.0	0.0

122410 rows × 96 columns

In [35]:
```
pca=PCA(n_components=84)
pca.fit(X)
```

Out[35]:
```
PCA(n_components=84)
```

In [38]:
```
plt.figure(figsize=(10,10))
plt.hist(pca.explained_variance_ratio_,bins=84)
plt.show()
```



In [39]:
```
def getKComponents(pca,alpha):
    total_variance =0

    for feature, variance in enumerate (pca.explained_variance_ratio_):

        total_variance +=variance
        if(total_variance >=1 -alpha):
            return feature +1
    return len(pca.explained_variance_ratio_)
```

In [40]:
```
K=getKComponents(pca, 0.05)
```

In [41]:
```
X=pca.transform(X)[:, 0:K]
pd.DataFrame(X)
```

Out[41]:

	0	1	2	3	4	5	6	7	8	9	...	23	24	25	26	27	28	29	30	31	32
0	0.807650	-3.041170	-0.477923	-0.788150	-0.837791	0.305314	-0.571604	-0.640431	0.391012	-0.089662	...	-0.406736	-0.220961	0.062492	-0.033852	-0.065296	0.007777	-0.080869	-0.049944	-0.165961	0.056775
1	0.403354	-2.516566	-0.271932	-1.210600	-0.940974	0.463845	-0.769151	-0.302018	-0.226065	-0.320548	...	-0.424848	0.001105	0.020463	0.051323	0.030929	-0.023390	0.045672	-0.006290	0.185634	0.072155
2	1.793312	-1.452247	-0.282613	-1.090443	-0.785376	0.333237	-0.704440	-0.376257	0.147196	-0.330615	...	0.156083	0.008170	-0.026589	0.010115	0.146338	-0.054307	0.084381	-0.020325	0.149910	0.107855
3	1.953921	-1.452769	-0.437852	-1.067899	-0.855280	0.367944	-0.569773	-0.317184	0.120949	-0.332918	...	0.102953	0.022223	0.006126	0.070663	0.069446	-0.048108	0.049091	-0.094075	0.219295	-0.199274
4	0.531003	-3.044842	-0.766788	-0.128865	-0.392524	0.306517	-0.459914	-0.600220	0.627068	-0.421338	...	-0.332748	-0.130514	-0.103789	0.085502	-0.007072	-0.034536	0.101459	-0.186271	0.057514	-0.024417
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
122405	2.662008	1.291910	1.542080	0.772222	-0.739458	-0.647647	0.687017	-0.644861	-1.184389	0.700592	...	0.002813	0.159939	0.097054	-0.472091	-0.097370	-0.511502	-0.017211	-0.110692	-0.203674	0.171138
122406	-0.110277	-2.209320	0.228796	1.364919	-0.437793	-0.861938	-0.019721	0.045146	-0.401692	0.860939	...	0.229384	0.662024	0.323117	-0.426443	-0.007367	-0.088607	0.094928	0.059359	-0.130469	0.205325
122407	-2.375611	0.942436	1.160162	-0.669273	1.450319	1.070238	0.582792	0.596464	-1.143166	1.541248	...	0.833349	0.424366	-0.131201	0.113948	-0.116364	-0.103917	0.191911	-0.500739	-0.255426	0.079432
122408	-1.905041	0.618990	1.723658	-0.066840	0.992627	1.524162	0.628238	0.485320	-1.258091	1.679712	...	0.700360	0.455182	-0.235973	0.224711	-0.127013	-0.221656	0.129971	-0.494348	-0.006301	-0.281792
122409	-0.038915	0.029125	2.258288	0.377764	0.608470	0.863123	0.803317	0.310871	-1.703889	0.727512	...	0.778485	0.338943	-0.033668	-0.103808	-0.136386	-0.163887	0.142128	-0.441898	0.023163	-0.362651

122410 rows × 33 columns

In [42]:
```
X_train, X_test, y_train, y_test =train_test_split(X,y, train_size=0.8)
```

In [43]:
```
log_model=LogisticRegression(verbose=True)
nn_model=MLPClassifier(verbose=True)


log_model.fit(X_train,y_train)
nn_model.fit(X_train,y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.8s finished
```

```
Iteration 1, loss = 0.48377887
Iteration 2, loss = 0.45807745
Iteration 3, loss = 0.45241656
Iteration 4, loss = 0.44863221
Iteration 5, loss = 0.44572142
Iteration 6, loss = 0.44349128
Iteration 7, loss = 0.44112138
Iteration 8, loss = 0.43914951
Iteration 9, loss = 0.43744838
Iteration 10, loss = 0.43558674
Iteration 11, loss = 0.43448623
Iteration 12, loss = 0.43284445
Iteration 13, loss = 0.43146943
Iteration 14, loss = 0.43038017
Iteration 15, loss = 0.42909181
Iteration 16, loss = 0.42778036
Iteration 17, loss = 0.42695717
Iteration 18, loss = 0.42592105
Iteration 19, loss = 0.42486446
Iteration 20, loss = 0.42399641
Iteration 21, loss = 0.42325562
Iteration 22, loss = 0.42233563
Iteration 23, loss = 0.42131125
Iteration 24, loss = 0.42007509
Iteration 25, loss = 0.41946778
Iteration 26, loss = 0.41872247
Iteration 27, loss = 0.41769155
Iteration 28, loss = 0.41699354
Iteration 29, loss = 0.41640880
Iteration 30, loss = 0.41544550
Iteration 31, loss = 0.41473680
Iteration 32, loss = 0.41382941
Iteration 33, loss = 0.41317603
Iteration 34, loss = 0.41216222
Iteration 35, loss = 0.41189421
Iteration 36, loss = 0.41098714
Iteration 37, loss = 0.41013890
Iteration 38, loss = 0.40984341
Iteration 39, loss = 0.40897038
Iteration 40, loss = 0.40819408
Iteration 41, loss = 0.40769576
Iteration 42, loss = 0.40706019
Iteration 43, loss = 0.40675026
Iteration 44, loss = 0.40609636
Iteration 45, loss = 0.40542777
Iteration 46, loss = 0.40516763
Iteration 47, loss = 0.40454577
Iteration 48, loss = 0.40387622
Iteration 49, loss = 0.40317883
Iteration 50, loss = 0.40305602
Iteration 51, loss = 0.40295693
Iteration 52, loss = 0.40191145
Iteration 53, loss = 0.40194830
Iteration 54, loss = 0.40148198
Iteration 55, loss = 0.40095124
Iteration 56, loss = 0.40028403
Iteration 57, loss = 0.40039476
Iteration 58, loss = 0.39950964
Iteration 59, loss = 0.39959462
Iteration 60, loss = 0.39937248
Iteration 61, loss = 0.39813674
Iteration 62, loss = 0.39818059
Iteration 63, loss = 0.39806733
Iteration 64, loss = 0.39773215
Iteration 65, loss = 0.39719919
Iteration 66, loss = 0.39704536
Iteration 67, loss = 0.39671242
Iteration 68, loss = 0.39640389
Iteration 69, loss = 0.39633540
Iteration 70, loss = 0.39558899
Iteration 71, loss = 0.39561688
Iteration 72, loss = 0.39505491
Iteration 73, loss = 0.39461184
Iteration 74, loss = 0.39467400
Iteration 75, loss = 0.39424257
Iteration 76, loss = 0.39407967
Iteration 77, loss = 0.39410974
Iteration 78, loss = 0.39328831
Iteration 79, loss = 0.39285190
Iteration 80, loss = 0.39327856
Iteration 81, loss = 0.39243713
Iteration 82, loss = 0.39302224
Iteration 83, loss = 0.39245893
Iteration 84, loss = 0.39201591
Iteration 85, loss = 0.39173846
Iteration 86, loss = 0.39129107
Iteration 87, loss = 0.39150073
Iteration 88, loss = 0.39145271
Iteration 89, loss = 0.39079655
Iteration 90, loss = 0.39094115
Iteration 91, loss = 0.39065704
Iteration 92, loss = 0.39051831
Iteration 93, loss = 0.39041181
Iteration 94, loss = 0.38998537
Iteration 95, loss = 0.38995393
Iteration 96, loss = 0.38969698
Iteration 97, loss = 0.38954285
Iteration 98, loss = 0.38947948
Iteration 99, loss = 0.38916578
Iteration 100, loss = 0.38874534
Iteration 101, loss = 0.38858772
Iteration 102, loss = 0.38839324
Iteration 103, loss = 0.38835531
Iteration 104, loss = 0.38809775
Iteration 105, loss = 0.38764416
Iteration 106, loss = 0.38766486
Iteration 107, loss = 0.38819617
Iteration 108, loss = 0.38745094
Iteration 109, loss = 0.38722433
Iteration 110, loss = 0.38723997
Iteration 111, loss = 0.38708954
Iteration 112, loss = 0.38701594
Iteration 113, loss = 0.38672314
Iteration 114, loss = 0.38648242
Iteration 115, loss = 0.38633833
Iteration 116, loss = 0.38597799
Iteration 117, loss = 0.38577618
Iteration 118, loss = 0.38615824
Iteration 119, loss = 0.38587155
Iteration 120, loss = 0.38596808
Iteration 121, loss = 0.38548930
Iteration 122, loss = 0.38537097
Iteration 123, loss = 0.38543392
Iteration 124, loss = 0.38521417
Iteration 125, loss = 0.38555165
Iteration 126, loss = 0.38518150
Iteration 127, loss = 0.38460043
Iteration 128, loss = 0.38445482
Iteration 129, loss = 0.38430146
Iteration 130, loss = 0.38467462
Iteration 131, loss = 0.38414992
Iteration 132, loss = 0.38484714
Iteration 133, loss = 0.38430841
```

```
Iteration 134, loss = 0.38421945
Iteration 135, loss = 0.38414827
Iteration 136, loss = 0.38423717
Iteration 137, loss = 0.38378947
Iteration 138, loss = 0.38407505
Iteration 139, loss = 0.38391970
Iteration 140, loss = 0.38343937
Iteration 141, loss = 0.38391945
Iteration 142, loss = 0.38320225
Iteration 143, loss = 0.38376077
Iteration 144, loss = 0.38332218
Iteration 145, loss = 0.38348385
Iteration 146, loss = 0.38286923
Iteration 147, loss = 0.38341340
Iteration 148, loss = 0.38308038
Iteration 149, loss = 0.38236077
Iteration 150, loss = 0.38265548
Iteration 151, loss = 0.38258832
Iteration 152, loss = 0.38242038
Iteration 153, loss = 0.38242348
Iteration 154, loss = 0.38226832
Iteration 155, loss = 0.38219453
Iteration 156, loss = 0.38205031
Iteration 157, loss = 0.38235196
Iteration 158, loss = 0.38187359
Iteration 159, loss = 0.38122819
Iteration 160, loss = 0.38147142
Iteration 161, loss = 0.38160972
Iteration 162, loss = 0.38125877
Iteration 163, loss = 0.38146518
Iteration 164, loss = 0.38159088
Iteration 165, loss = 0.38157955
Iteration 166, loss = 0.38115770
Iteration 167, loss = 0.38092241
Iteration 168, loss = 0.38100067
Iteration 169, loss = 0.38094555
Iteration 170, loss = 0.38087988
Iteration 171, loss = 0.38064960
Iteration 172, loss = 0.38059221
Iteration 173, loss = 0.38091446
Iteration 174, loss = 0.38101576
Iteration 175, loss = 0.38064505
Iteration 176, loss = 0.38032782
Iteration 177, loss = 0.38091042
Iteration 178, loss = 0.38029796
Iteration 179, loss = 0.38099635
Iteration 180, loss = 0.38027215
Iteration 181, loss = 0.38011015
Iteration 182, loss = 0.38009929
Iteration 183, loss = 0.37991770
Iteration 184, loss = 0.38003867
Iteration 185, loss = 0.37971281
Iteration 186, loss = 0.37994869
Iteration 187, loss = 0.37941748
Iteration 188, loss = 0.37987124
Iteration 189, loss = 0.38032852
Iteration 190, loss = 0.37948062
Iteration 191, loss = 0.38037455
Iteration 192, loss = 0.37976596
Iteration 193, loss = 0.37928084
Iteration 194, loss = 0.37924651
Iteration 195, loss = 0.37950517
Iteration 196, loss = 0.37961234
Iteration 197, loss = 0.37973104
Iteration 198, loss = 0.37913674
Iteration 199, loss = 0.37908003
Iteration 200, loss = 0.37938997
```

```
D:\anaconda\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[43]:
```
MLPClassifier(verbose=True)
```

In [45]:
```
print(f"Logistic Model : {log_model.score(X_test,y_test)}")
print(f"Neural Net Model: {nn_model.score(X_test,y_test)}")
```

```
Logistic Model : 0.745282248182338
Neural Net Model: 0.7824524140184625
```

In [ ]: