In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
#try within same table encoding
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn import svm
```

In [2]:
```
iris = pd.read_csv('Iris.csv')
```

In [3]:
```
print(iris)
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

In [4]:
```
iris.head()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [5]:
```
irisallvisual = iris
```

In [6]:
```
iris.isnull().sum()
```

Out[6]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [7]:
```
# encodecols= ['Species']

# for col in encodecols:
#     iris[col]=le.fit_transform(iris[col])
#     print(le.classes_)

# iris.head(5)
```

In [8]:
```
#  MinMaxScaler
mms = MinMaxScaler()
```

In [9]:
```
# fit scaler  napravila između 0 i 1
iris[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']] = mms.fit_transform(iris[['SepalLengthCm', 'Se
iris.head(10)
```

Out[9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	0.222222	0.625000	0.067797	0.041667	Iris-setosa
1	2	0.166667	0.416667	0.067797	0.041667	Iris-setosa
2	3	0.111111	0.500000	0.050847	0.041667	Iris-setosa
3	4	0.083333	0.458333	0.084746	0.041667	Iris-setosa
4	5	0.194444	0.666667	0.067797	0.041667	Iris-setosa
5	6	0.305556	0.791667	0.118644	0.125000	Iris-setosa
6	7	0.083333	0.583333	0.067797	0.083333	Iris-setosa
7	8	0.194444	0.583333	0.084746	0.041667	Iris-setosa
8	9	0.027778	0.375000	0.067797	0.041667	Iris-setosa
9	10	0.166667	0.458333	0.084746	0.000000	Iris-setosa

In [10]:
```
# from sklearn.preprocessing import StandardScaler, example why not standard scaler
# scaler = StandardScaler()
# iris[['SepalLengthCm', 'SepalWidthCm','PetalLengthCm', 'PetalWidthCm']] = scaler.fit_transform(iris[['SepalLengthCm
# iris.head(10)
```

In [11]:
```
# Y=iris.iloc[:,5]
# print(Y)
#svi redovi i samo prva kolonoa tj nul kolona
```

In [12]:
```
# X=iris.iloc[:,1:].drop(columns=['Species'])
# print(X)
# #svi redovi i sve kolone osim prve id
```

In [13]:
```

# from sklearn.model_selection import train_test_split
```

In [14]:
```
# X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.30)
```

In [15]:
```
# print(Y_train.shape)
# print(Y_test.shape)
```

In [16]:
```
# from sklearn.neighbors import KNeighborsClassifier
# clfknn = KNeighborsClassifier(n_neighbors = 8)
# clfknn.fit(X_train, Y_train)
```

In [17]:
```
# yhat = clfknn.predict(X_test)
```

In [18]:
```
# print(yhat)
#prediktivni rezulTati 30% izdvojenih stavki X_testa
```

In [19]:
```
# usporeda prediktivnih i stvarnih rezultata
# accuracy_score(Y_test, yhat)
```

In [20]:
```
#  clfsvm = svm.SVC()
```

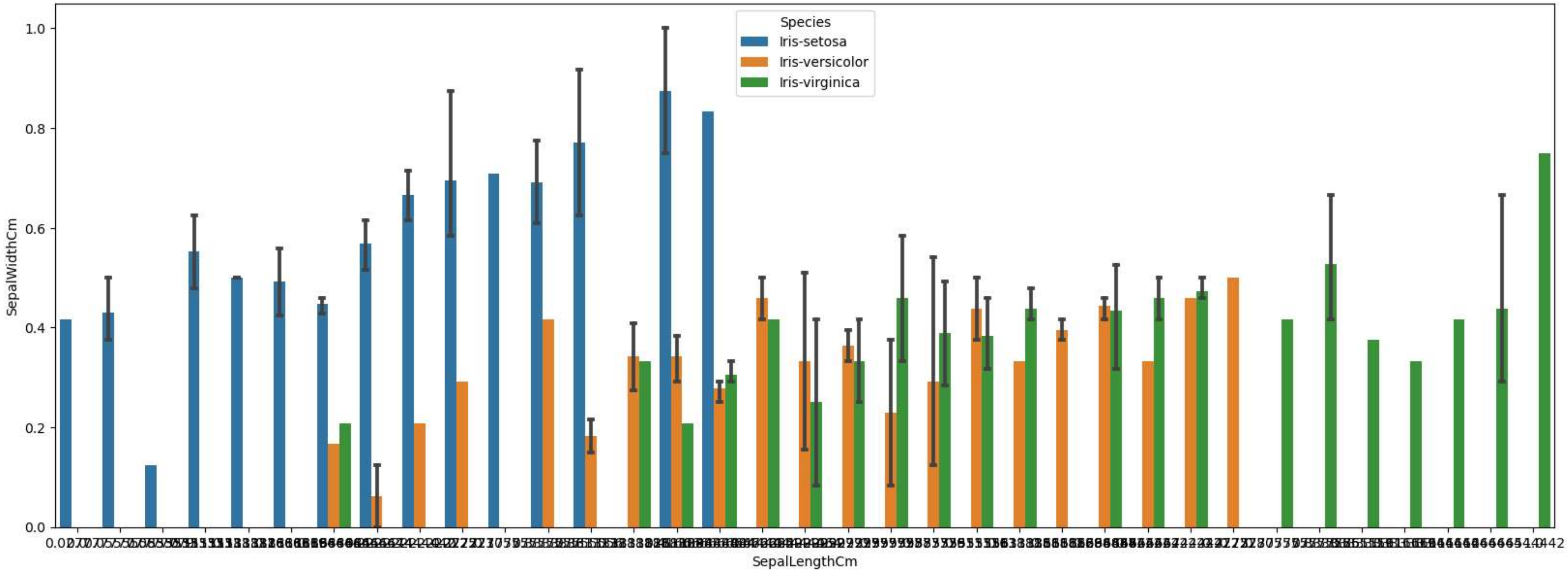
In [21]:
```
# clfsvm.fit(X_train, Y_train)
```

In [22]:
```
# yhatSVM = clfsvm.predict(X_test)
```

In [23]:
```
# print(yhatSVM)
```

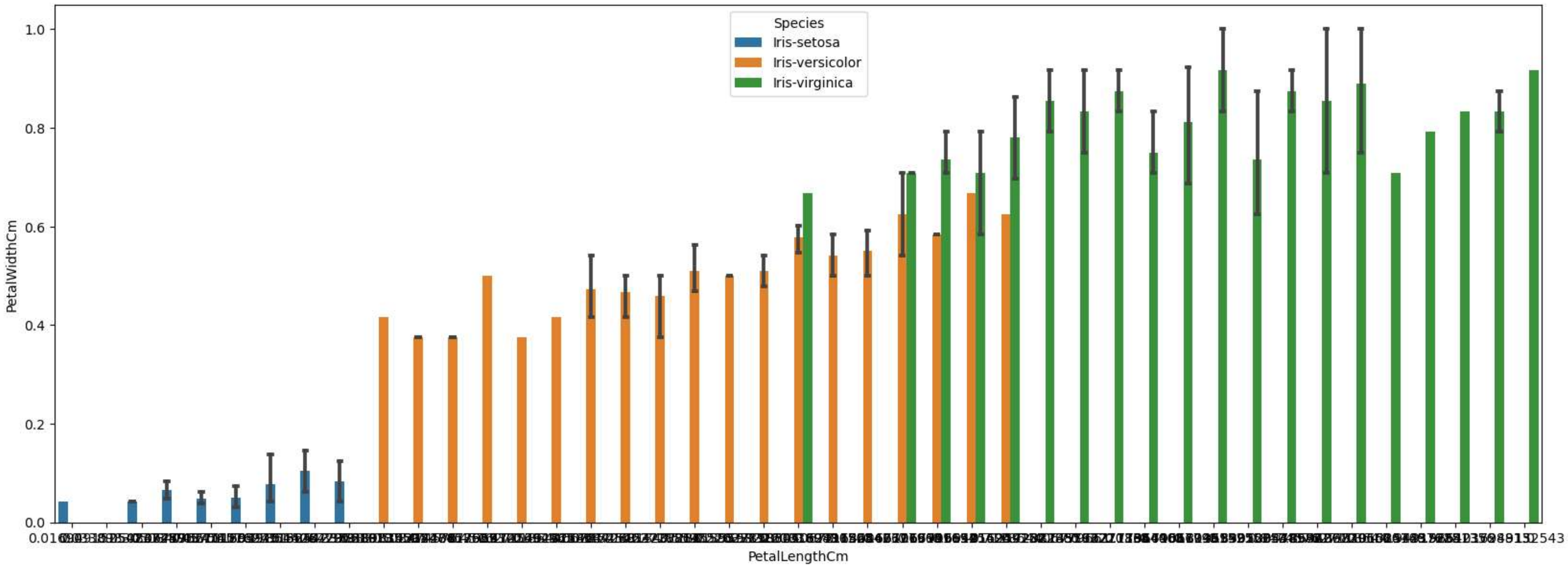
In [24]:
```
#usporeda prediktivnih i stvarnih rezultata
# accuracy_score(Y_test, yhatSVM)
```

In [25]:
```
fig, ax = plt.subplots(figsize = (20,7))
sns.barplot(x='SepalLengthCm', y='SepalWidthCm', hue='Species', capsize=0.09, data=irisallvisual)
plt.pause(0.001)
plt.show
# import matplotlib.pyplot as plt

# iris.plot(x=['SepalLengthCm', 'SepalLengthCm', 'PetalLengthCm', 'PetalWidthCm'], y='Species', kind="bar")
```



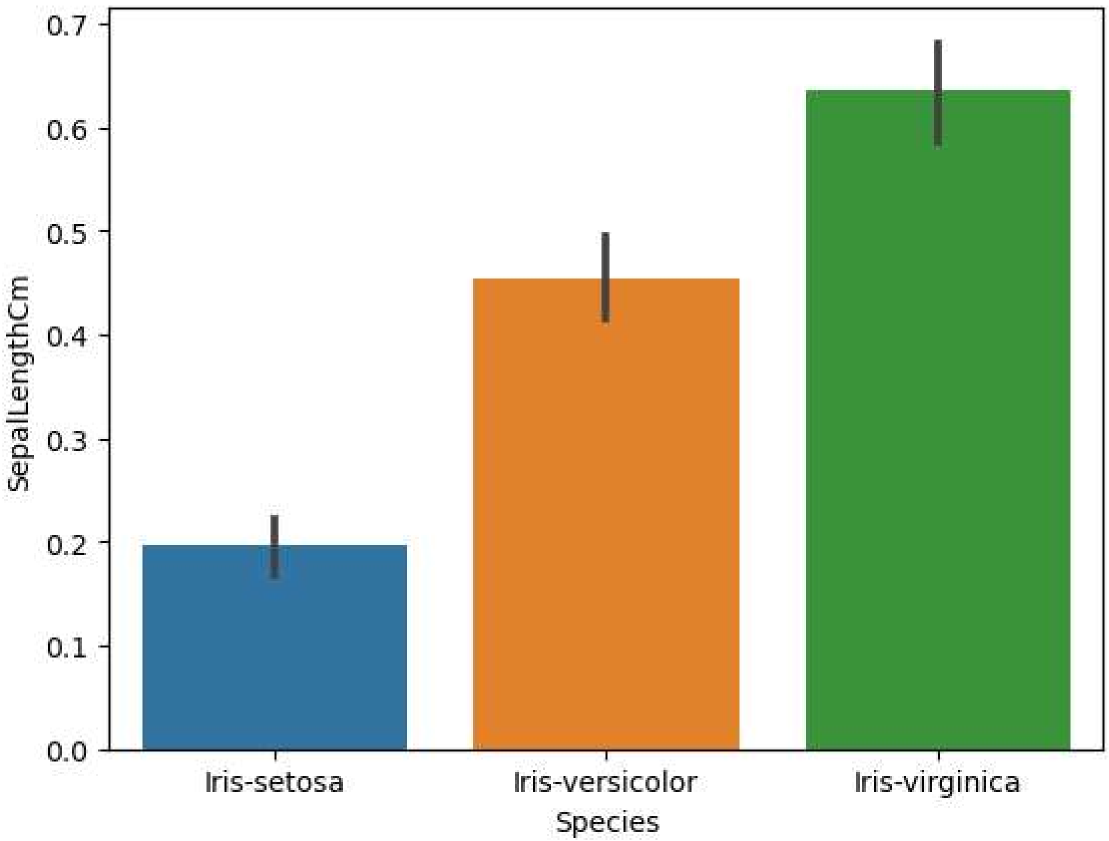
Out[25]: <function matplotlib.pyplot.show(close=None, block=None)>

In [26]:
```
fig, ax = plt.subplots(figsize = (20,7))
sns.barplot(x='PetalLengthCm', y='PetalWidthCm', hue='Species', capsize=0.09, data=irisallvisual)
plt.pause(0.001)
plt.show
```



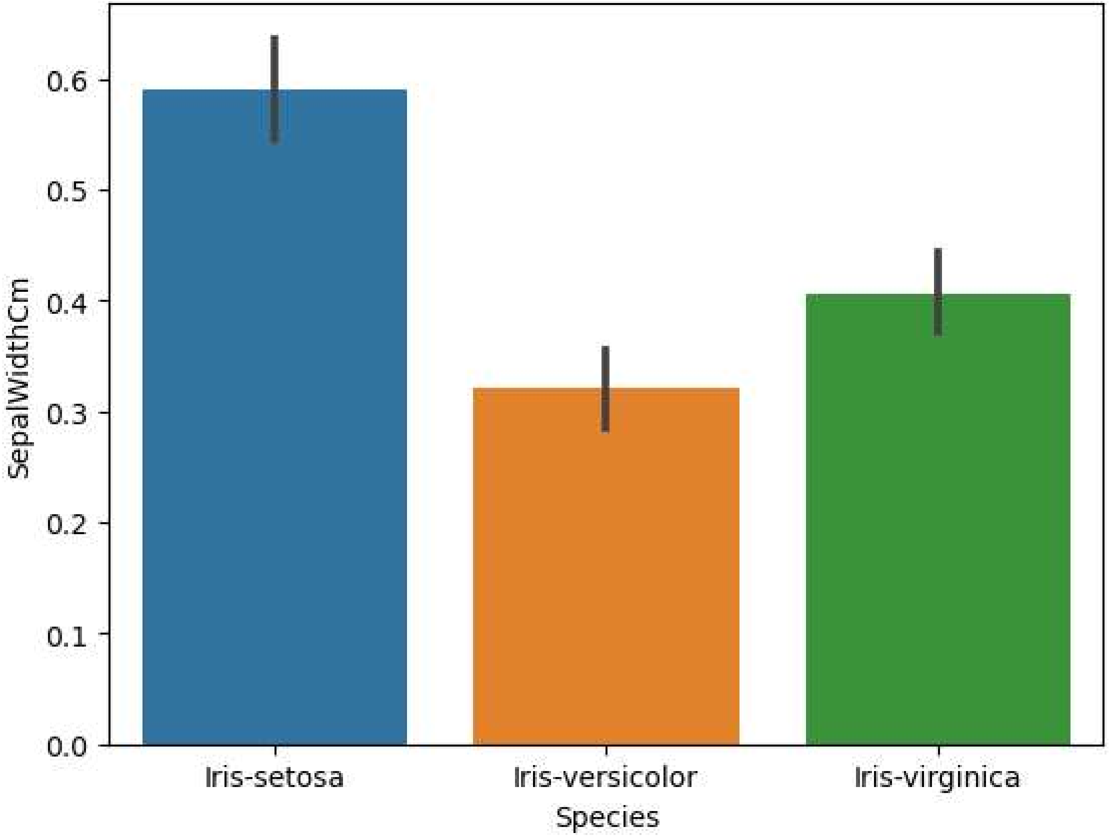
Out[26]: <function matplotlib.pyplot.show(close=None, block=None)>

In [27]:
```
k = irisallvisual.groupby(['Species','SepalLengthCm'])['SepalLengthCm'].agg(['mean']).reset_index()
sns.barplot(y='SepalLengthCm', x= 'Species', data = irisallvisual)
plt.xlabel('Species', fontsize=10)
plt.ylabel('SepalLengthCm', fontsize=10)
plt.pause(0.001)
plt.show
```



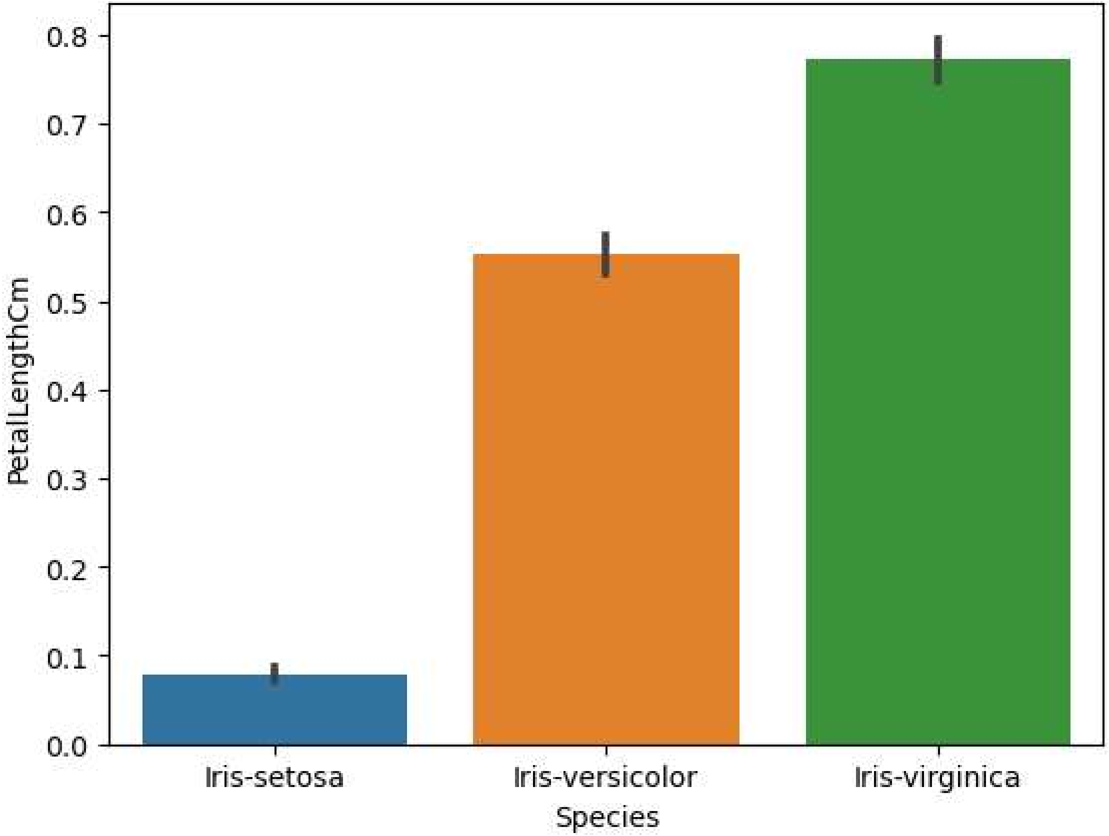
Out[27]: <function matplotlib.pyplot.show(close=None, block=None)>

In [28]:
```
l = irisallvisual.groupby(['Species', 'SepalWidthCm'])['SepalWidthCm'].agg(['mean']).reset_index()
sns.barplot(y='SepalWidthCm', x= 'Species', data = irisallvisual)
plt.xlabel('Species', fontsize=10)
plt.ylabel('SepalWidthCm', fontsize=10)
plt.pause(0.001)
plt.show
```



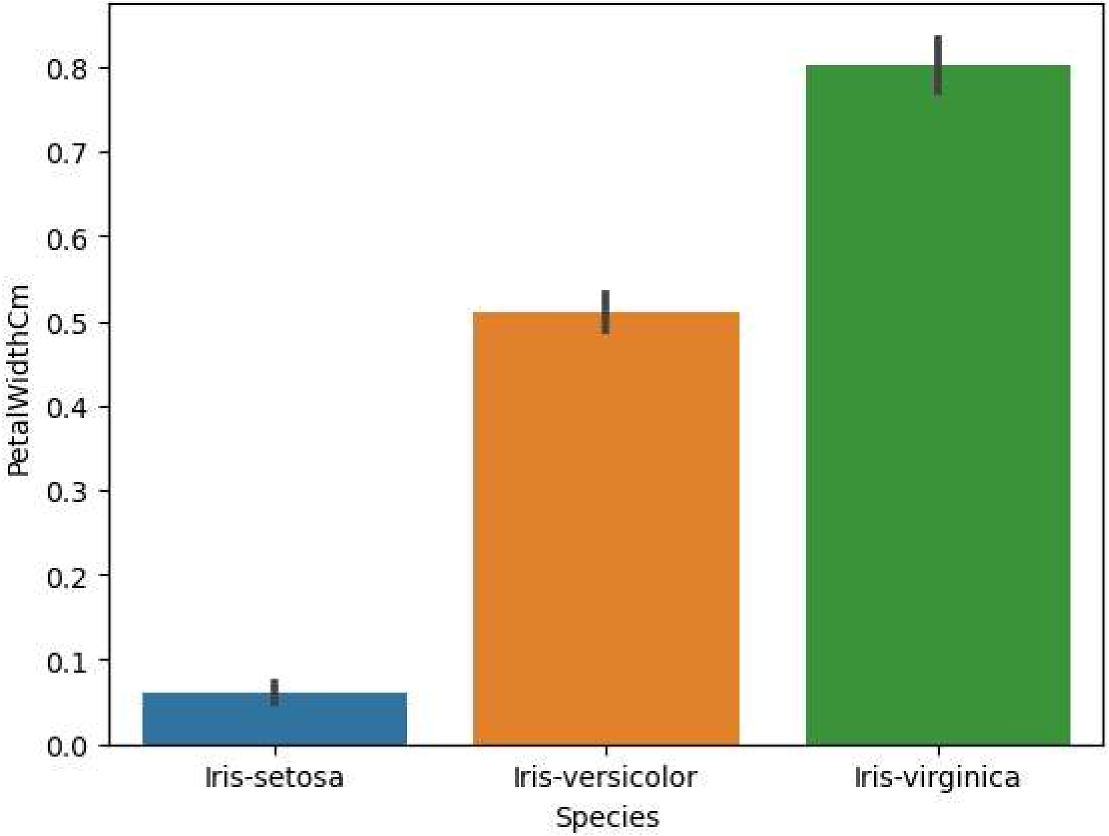
Out[28]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [29]:
```
m = irisallvisual.groupby(['Species','PetalLengthCm'])['PetalLengthCm'].agg(['mean']).reset_index()
sns.barplot(y='PetalLengthCm', x= 'Species', data = irisallvisual)
plt.xlabel('Species', fontsize=10)
plt.ylabel('PetalLengthCm', fontsize=10)
plt.pause(0.001)
plt.show
```



Out[29]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [30]:
```
n = irisallvisual.groupby(['Species','PetalWidthCm'])['PetalWidthCm'].agg(['mean']).reset_index()
sns.barplot(y='PetalWidthCm', x= 'Species', data = irisallvisual)
plt.xlabel('Species', fontsize=10)
plt.ylabel('PetalWidthCm', fontsize=10)
plt.pause(0.001)
plt.show
```



Out[30]: `<function matplotlib.pyplot.show(close=None, block=None)>`

In [ ]:

In [ ]:

In [31]:

```
IRIS_SETOSA = []

for i in iris['Species']:
    if 'setosa' in i:
        IRIS_SETOSA.append(1)
    else:
        IRIS_SETOSA.append(0)
```

In [32]:
```
iris.head(55)
```

Out[32]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	0.222222	0.625000	0.067797	0.041667	Iris-setosa
1	2	0.166667	0.416667	0.067797	0.041667	Iris-setosa
2	3	0.111111	0.500000	0.050847	0.041667	Iris-setosa
3	4	0.083333	0.458333	0.084746	0.041667	Iris-setosa
4	5	0.194444	0.666667	0.067797	0.041667	Iris-setosa
5	6	0.305556	0.791667	0.118644	0.125000	Iris-setosa
6	7	0.083333	0.583333	0.067797	0.083333	Iris-setosa
7	8	0.194444	0.583333	0.084746	0.041667	Iris-setosa
8	9	0.027778	0.375000	0.067797	0.041667	Iris-setosa
9	10	0.166667	0.458333	0.084746	0.000000	Iris-setosa
10	11	0.305556	0.708333	0.084746	0.041667	Iris-setosa
11	12	0.138889	0.583333	0.101695	0.041667	Iris-setosa
12	13	0.138889	0.416667	0.067797	0.000000	Iris-setosa
13	14	0.000000	0.416667	0.016949	0.000000	Iris-setosa
14	15	0.416667	0.833333	0.033898	0.041667	Iris-setosa
15	16	0.388889	1.000000	0.084746	0.125000	Iris-setosa
16	17	0.305556	0.791667	0.050847	0.125000	Iris-setosa
17	18	0.222222	0.625000	0.067797	0.083333	Iris-setosa
18	19	0.388889	0.750000	0.118644	0.083333	Iris-setosa
19	20	0.222222	0.750000	0.084746	0.083333	Iris-setosa
20	21	0.305556	0.583333	0.118644	0.041667	Iris-setosa
21	22	0.222222	0.708333	0.084746	0.125000	Iris-setosa
22	23	0.083333	0.666667	0.000000	0.041667	Iris-setosa
23	24	0.222222	0.541667	0.118644	0.166667	Iris-setosa
24	25	0.138889	0.583333	0.152542	0.041667	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
25	26	0.194444	0.416667	0.101695	0.041667	Iris-setosa
26	27	0.194444	0.583333	0.101695	0.125000	Iris-setosa
27	28	0.250000	0.625000	0.084746	0.041667	Iris-setosa
28	29	0.250000	0.583333	0.067797	0.041667	Iris-setosa
29	30	0.111111	0.500000	0.101695	0.041667	Iris-setosa
30	31	0.138889	0.458333	0.101695	0.041667	Iris-setosa
31	32	0.305556	0.583333	0.084746	0.125000	Iris-setosa
32	33	0.250000	0.875000	0.084746	0.000000	Iris-setosa
33	34	0.333333	0.916667	0.067797	0.041667	Iris-setosa
34	35	0.166667	0.458333	0.084746	0.000000	Iris-setosa
35	36	0.194444	0.500000	0.033898	0.041667	Iris-setosa
36	37	0.333333	0.625000	0.050847	0.041667	Iris-setosa
37	38	0.166667	0.458333	0.084746	0.000000	Iris-setosa
38	39	0.027778	0.416667	0.050847	0.041667	Iris-setosa
39	40	0.222222	0.583333	0.084746	0.041667	Iris-setosa
40	41	0.194444	0.625000	0.050847	0.083333	Iris-setosa
41	42	0.055556	0.125000	0.050847	0.083333	Iris-setosa
42	43	0.027778	0.500000	0.050847	0.041667	Iris-setosa
43	44	0.194444	0.625000	0.101695	0.208333	Iris-setosa
44	45	0.222222	0.750000	0.152542	0.125000	Iris-setosa
45	46	0.138889	0.416667	0.067797	0.083333	Iris-setosa
46	47	0.222222	0.750000	0.101695	0.041667	Iris-setosa
47	48	0.083333	0.500000	0.067797	0.041667	Iris-setosa
48	49	0.277778	0.708333	0.084746	0.041667	Iris-setosa
49	50	0.194444	0.541667	0.067797	0.041667	Iris-setosa
50	51	0.750000	0.500000	0.627119	0.541667	Iris-versicolor

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
51	52	0.583333	0.500000	0.593220	0.583333	Iris-versicolor
52	53	0.722222	0.458333	0.661017	0.583333	Iris-versicolor
53	54	0.333333	0.125000	0.508475	0.500000	Iris-versicolor
54	55	0.611111	0.333333	0.610169	0.583333	Iris-versicolor

In [33]:
```
print(IRIS_SETOSA)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [34]:
```

iris['Species'] = IRIS_SETOSA
```

In [35]:
```
iris.head(51)
```

Out[35]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	0.222222	0.625000	0.067797	0.041667	1
1	2	0.166667	0.416667	0.067797	0.041667	1
2	3	0.111111	0.500000	0.050847	0.041667	1
3	4	0.083333	0.458333	0.084746	0.041667	1
4	5	0.194444	0.666667	0.067797	0.041667	1
5	6	0.305556	0.791667	0.118644	0.125000	1
6	7	0.083333	0.583333	0.067797	0.083333	1
7	8	0.194444	0.583333	0.084746	0.041667	1
8	9	0.027778	0.375000	0.067797	0.041667	1
9	10	0.166667	0.458333	0.084746	0.000000	1
10	11	0.305556	0.708333	0.084746	0.041667	1
11	12	0.138889	0.583333	0.101695	0.041667	1
12	13	0.138889	0.416667	0.067797	0.000000	1
13	14	0.000000	0.416667	0.016949	0.000000	1
14	15	0.416667	0.833333	0.033898	0.041667	1
15	16	0.388889	1.000000	0.084746	0.125000	1
16	17	0.305556	0.791667	0.050847	0.125000	1
17	18	0.222222	0.625000	0.067797	0.083333	1
18	19	0.388889	0.750000	0.118644	0.083333	1
19	20	0.222222	0.750000	0.084746	0.083333	1
20	21	0.305556	0.583333	0.118644	0.041667	1
21	22	0.222222	0.708333	0.084746	0.125000	1
22	23	0.083333	0.666667	0.000000	0.041667	1
23	24	0.222222	0.541667	0.118644	0.166667	1
24	25	0.138889	0.583333	0.152542	0.041667	1

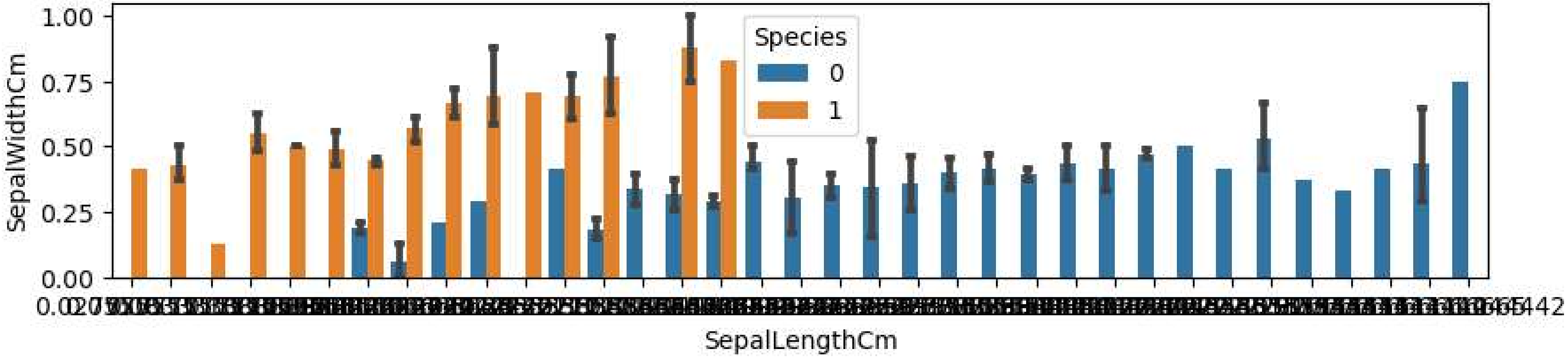
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
25	26	0.194444	0.416667	0.101695	0.041667	1
26	27	0.194444	0.583333	0.101695	0.125000	1
27	28	0.250000	0.625000	0.084746	0.041667	1
28	29	0.250000	0.583333	0.067797	0.041667	1
29	30	0.111111	0.500000	0.101695	0.041667	1
30	31	0.138889	0.458333	0.101695	0.041667	1
31	32	0.305556	0.583333	0.084746	0.125000	1
32	33	0.250000	0.875000	0.084746	0.000000	1
33	34	0.333333	0.916667	0.067797	0.041667	1
34	35	0.166667	0.458333	0.084746	0.000000	1
35	36	0.194444	0.500000	0.033898	0.041667	1
36	37	0.333333	0.625000	0.050847	0.041667	1
37	38	0.166667	0.458333	0.084746	0.000000	1
38	39	0.027778	0.416667	0.050847	0.041667	1
39	40	0.222222	0.583333	0.084746	0.041667	1
40	41	0.194444	0.625000	0.050847	0.083333	1
41	42	0.055556	0.125000	0.050847	0.083333	1
42	43	0.027778	0.500000	0.050847	0.041667	1
43	44	0.194444	0.625000	0.101695	0.208333	1
44	45	0.222222	0.750000	0.152542	0.125000	1
45	46	0.138889	0.416667	0.067797	0.083333	1
46	47	0.222222	0.750000	0.101695	0.041667	1
47	48	0.083333	0.500000	0.067797	0.041667	1
48	49	0.277778	0.708333	0.084746	0.041667	1
49	50	0.194444	0.541667	0.067797	0.041667	1
50	51	0.750000	0.500000	0.627119	0.541667	0

In [36]:
```
irisvisual = iris
irisvisual.head()
```

Out[36]:

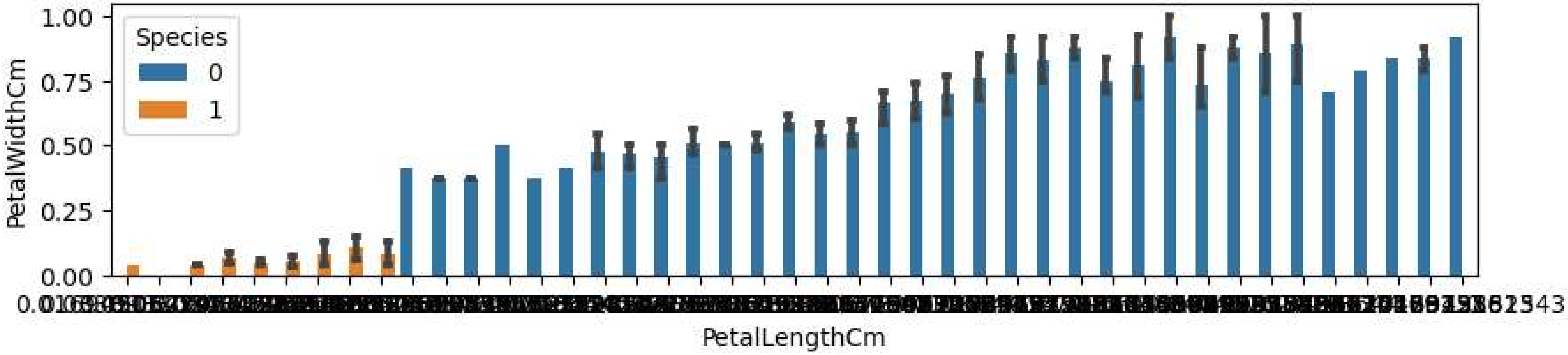
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	0.222222	0.625000	0.067797	0.041667	1
1	2	0.166667	0.416667	0.067797	0.041667	1
2	3	0.111111	0.500000	0.050847	0.041667	1
3	4	0.083333	0.458333	0.084746	0.041667	1
4	5	0.194444	0.666667	0.067797	0.041667	1

In [37]:
```
fig, ax = plt.subplots(figsize = (10,2))

sns.barplot(x='SepalLengthCm', y='SepalWidthCm', hue='Species', capsize=0.09, data=irisvisual)
plt.pause(0.001)
plt.show
#1 is Setosa, 0 is all other species of iris
# import matplotlib.pyplot as plt

# iris.plot(x=['SepalLengthCm', 'SepalLengthCm', 'PetalLengthCm', 'PetalWidthCm'], y='Species', kind="bar")
```



Out[37]: <function matplotlib.pyplot.show(close=None, block=None)>

In [38]:
```
fig, ax = plt.subplots(figsize = (10,2))
sns.barplot(x='PetalLengthCm', y='PetalWidthCm', hue='Species', capsize=0.09, data=irisvisual)
plt.pause(0.001)
plt.show

#1 is Setosa, 0 is all other species of iris
```



Out[38]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [39]:
```
Y=iris.iloc[:,5]
print(Y)
#svi redovi i samo prva kolonoa tj nul kolona
```

```
0      1
1      1
2      1
3      1
4      1
      ..
145    0
146    0
147    0
148    0
149    0
Name: Species, Length: 150, dtype: int64
```

In [40]:
```
X=iris.iloc[:,1:].drop(columns=['Species'])
print(X)
#svi redovi i sve kolone osim prve id
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0         0.222222      0.625000       0.067797      0.041667
1         0.166667      0.416667       0.067797      0.041667
2         0.111111      0.500000       0.050847      0.041667
3         0.083333      0.458333       0.084746      0.041667
4         0.194444      0.666667       0.067797      0.041667
..             ...           ...            ...           ...
145       0.666667      0.416667       0.711864      0.916667
146       0.555556      0.208333       0.677966      0.750000
147       0.611111      0.416667       0.711864      0.791667
148       0.527778      0.583333       0.745763      0.916667
149       0.444444      0.416667       0.694915      0.708333

[150 rows x 4 columns]
```

In [ ]:

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20)
```

In [43]:
```
print(Y_train.shape)
print(Y_test.shape)
```

```
(120,)
(30,)
```

In [44]:
```
#Species sad predstavlja ili je ili nije setosa..it is or isnt setosa
#bacause i want to show logistic regression example and permutation importance on sigmoid


from sklearn.linear_model import LogisticRegression
```

In [45]:
```
logreg = LogisticRegression(random_state=0, max_iter=1000)
logreg.fit(X_train, Y_train)
#primjenjuje nauceno iz traina i bez da na varijabli
```

Out[45]:
```
LogisticRegression(max_iter=1000, random_state=0)
```

In [46]:
```
yhat = logreg.predict(X_test)
```

In [47]:
```
accuracy_score(Y_test, yhat)
```

Out[47]:
```
1.0
```

In [48]:
```
#i will  do a permutation importance for features in logistic regression
#since all the features are numeric, I will use sigmoid function formula
#with euler method

#radit ću po formuli
#z = w0 + w1x1+ w2x2+ w3x3 + w4x4

#y = 1 / (1 + e-z)

#when x1 as first column and feature increases by 1 result is e^w1, same for others because
```

In [49]:
```
#********example from https://sefiks.com/2021/01/06/feature-importance-in-logistic-regression/#:~:
#text=The%20main%20difference%20between%20those,feature%20importance%20in%20logistic%20regression
#and
#https://stackoverflow.com/questions/74991339/how-to-plot-high-variance-feature-importance
#-values-generated-after-using-logist
# P(y = 1) / P(y = 0) = P(y = 1) / (1 – P(y = 1))

# Remember that we express the probability with logistic function

# P(y = 1) / (1 – P(y = 1)) = [ 1 / (1 + e-z) ] / [1 – (1 / (1 + e-z))]

# P(y = 1) / (1 – P(y = 1)) = [ 1 / (1 + e-z) ] / [(1 + e-z – 1) / (1 + e-z)] = 1 / e-z = e+z

# Let’s put the z term in the equation

# P(y = 1) / P(y = 0) = e^(w0 + w1x1+ w2x2+ w3x3 + w4x4)

# BTW, we call the left side of this equation odds.

# Let’s focus on a specific feature. E.g. x3. What happens to prediction when you make a change on x3 by 1 unit.
#I mean that I will change x3 to (x3 + 1). This is very similar to the definition of derivative.

# y3new / y3 = e^(w0 + w1x1+ w2x2+ w3(x3+1) + w4x4) / e^(w0 + w1x1+ w2x2+ w3x3 + w4x4)

# Remember that e^a / e^b = e^(a-b). I will apply this rule to the equation above.

# y3new / y3= e^(w0 + w1x1+ w2x2+ w3(x3+1) + w4x4 – (w0 + w1x1+ w2x2+ w3x3 + w4x4))

# y3new / y3 = e^(w0 + w1x1+ w2x2+ w3(x3+1) + w4x4 – w0 – w1x1– w2x2 – w3x3 – w4x4)

# y3new / y3 = e^(w3(x3+1) – w3x3) = e^(w3x3+w3 – w3x3)

# y3new / y3 = ew3
```

In [50]:
```
logreg.intercept_
```

Out[50]:
```
array([1.3567343])
```

In [51]:
```
logreg.intercept_[0]
#da ne bude array
```

Out[51]:
```
1.356734300318374
```

In [52]:
```
logreg.coef_
```

Out[52]:
```
array([[-1.46641201,  1.75523176, -3.06234545, -3.02379186]])
```

In [53]:
```
 w1, w2, w3, w4 = logreg.coef_[0]
```

In [54]:
```
w1, w2, w3, w4
```

Out[54]:
```
(-1.4664120099883533,
 1.7552317640729191,
 -3.0623454514278743,
 -3.0237918563184683)
```

In [55]:
```
#e ≈ 2.71828

#uzet cu test dio tablice, povećati svaku kolonu, feature za1, ostale ostavit same, gledamo što se dogodi funkciji tj
#iznosi e^w1; ako imam e i imam w1 u redu iznad iz logistickog modela te koeficijente
#samo se gleda snaga utjecaja na cijelu log funkciju odnosno utjecaj promjene svakog featuresa y1 na y u njihovom odno
#y u njihovom odnosu, a 1 ili 0 uvijek, al kako x-evi numerici mogu gledat povećanje za 1 ili sl.
#pa mogu i gledat brojcani utjecaj
#korak za x1 za 1 povećanje povećava y za y stari plud ynovi, koji je odnos dela x-a i kua odnosno derivacije funkcije
#u točki x1
# procjenjuje se vjerojatnost da je y novi =1 iu odnosu da nije 1 tj da je 0 i sve to u odnosu vjerojatnosti da je y 1
#u odnosu da y nije 1, tj da je 0, jer 1 razlomak p1 kroz p0 za y cini e na w ove sve..
# z je ka linearna jednadzba a y iz toga izvedena logisticka krivulja uz eulreov broj e

#kut je derivacija funkcije u točki x, odnosno nagib tangente na funkciju u točki x pa mogu korak po korak kontat pomo
#uzduz logisticke funkcije
```

In [56]:
```
w = [w1, w2, w3, w4]
```

In [57]:
```
import math
feature_names = ['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
#i will create a dataframe with columns Features and Importance
feature_importance=pd.DataFrame(X_test.columns, columns=['Features'])
feature_importance['Importance']=pow(math.e,np.array(w))
feature_importance.sort_values(by=['Importance'],ascending=False).reset_index()
```

Out[57]:

	index	Features	Importance
0	1	SepalWidthCm	5.784788
1	0	SepalLengthCm	0.230752
2	3	PetalWidthCm	0.048617
3	2	PetalLengthCm	0.046778

In [58]:
```
#sepal width ima najjaci utjecaj na cinjenicu je li iris setosa ili nije
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: