

Single layer perceptron

Unsupported Cell Type. Double-Click to inspect/edit the content.

Start coding or generate with AI.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('suv dataset.csv')
df.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
df.shape
```

```
(400, 5)
```

```
for col in df.columns:
    print(f"Unique values in '{col}': {df[col].unique()}")
```

```
Unique values in 'User ID': [15624510 15810944 15668575 15603246 15804002 15728773 15598044 15694829
15600575 15727311 15570769 15606274 15746139 15704987 15628972 15697686
15733883 15617482 15704583 15621083 15649487 15736760 15714658 15599081
15705113 15631159 15792818 15633531 15744529 15669656 15581198 15729054
15573452 15776733 15724858 15713144 15690188 15689425 15671766 15782806
15764419 15591915 15772798 15792008 15715541 15639277 15798850 15776348
15727696 15793813 15694395 15764195 15744919 15671655 15654901 15649136
15775562 15807481 15642885 15789109 15814004 15673619 15595135 15583681
15605000 15718071 15679760 15654574 15577178 15595324 15756932 15726358
15595228 15782530 15592877 15651983 15746737 15774179 15667265 15655123
15595917 15668385 15709476 15711218 15798659 15663939 15694946 15631912
15768816 15682268 15684881 15636428 15809823 15699284 15786993 15709441
15710257 15582492 15575694 15756820 15766289 15593014 15584545 15675949
15672091 15801658 15706185 15789863 15720943 15697997 15665416 15660200
15619653 15773447 15739160 15689237 15679297 15591433 15642725 15701962
15811613 15741049 15724423 15574305 15678168 15697020 15610801 15745232
15722758 15792102 15675185 15801247 15725660 15638963 15800061 15578006
15668504 15687491 15610403 15741094 15807909 15666141 15617134 15783029
15622833 15746422 15750839 15749130 15779862 15767871 15679651 15576219
15699247 15619087 15605327 15610140 15791174 15602373 15762605 15598840
15744279 15670619 15595933 15757837 15697574 15578738 15762228 15614827
15789815 15579781 15587013 15570932 15794661 15581654 15644296 15614420
15609653 15594577 15584114 15673367 15685576 15774727 15694288 15603319
15759666 15814816 15724402 15571059 15674206 15715160 15730448 15662067
15779581 15662901 15689751 15667742 15738448 15680243 15745083 15708228
15628523 15708196 15735549 15809347 15660866 15766609 15654230 15794566
15800890 15697424 15724536 15735878 15707596 15657163 15622478 15779529
15636023 15582066 15666675 15732987 15789432 15663161 15694879 15593715
15575002 15622171 15795224 15685346 15691808 15721007 15794253 15694453
15813113 15614187 15619407 15646227 15660541 15753874 15617877 15772073
15701537 15736228 15780572 15769596 15586996 15722061 15638003 15775590
15730688 15753102 15810075 15723373 15795298 15584320 15724161 15750056
15609637 15794493 15569641 15815236 15811177 15680587 15672821 15767681
15600379 15801336 15721592 15581282 15746203 15583137 15680752 15688172
15791373 15589449 15692819 15727467 15734312 15764604 15613014 15759684
15609669 15685536 15750447 15663249 15638646 15734161 15631070 15761950
15649668 15713912 15586757 15596522 15625395 15760570 15566689 15725794
15673539 15705298 156575791 15747043 15736397 15678201 15720745 15637593
15598070 15787556 15603942 15733973 15596761 15652400 15717893 15622585
15733964 15753861 15747097 15594762 15667417 15684861 15742204 15623502
15774872 15611191 15674331 15619465 15575247 15695679 15713463 15785170
15796351 15639576 15693264 15589715 15769902 15587177 15814553 15601550
15664907 15612465 15810800 15665760 15588080 15776844 15717560 15629739
15729908 15716781 15646936 15768151 15579212 15721835 15800515 15591279]
```

```

15587419 15750335 15699619 15606472 15778368 15671387 15573926 15709183
15577514 15778830 15768072 15768293 15654456 15807525 15574372 15671249
15779744 15624755 15611430 15774744 15629885 15708791 15793890 15646091
15596984 15800215 15577806 15749381 15683758 15670615 15715622 15707634
15806901 15775335 15724150 15627220 15672330 15668521 15807837 15592570
15748589 15635893 15757632 15691863 15706071 15654296 15755018 15594041]
Unique values in 'Gender': ['Male' 'Female']
Unique values in 'Age': [19 35 26 27 32 25 20 18 29 47 45 46 48 49 31 21 28 33 30 23 24 22 59 34
39 38 37 42 40 36 41 58 55 52 60 56 53 50 51 57 44 43 54]
Unique values in 'EstimatedSalary': [ 19000 20000 43000 57000 76000 58000 84000 150000 33000 65000
80000 52000 86000 18000 82000 25000 26000 28000 29000 22000
49000 41000 23000 30000 74000 137000 16000 44000 90000 27000
72000 31000 17000 51000 108000 15000 79000 54000 135000 89000
32000 83000 55000 48000 117000 87000 66000 120000 63000 68000

```

```
df.isnull().sum()
```

```

User ID      0
Gender       0
Age          0
EstimatedSalary 0
Purchased    0
dtype: int64

```

```
df.duplicated().sum()
```

```
0
```

```
df.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```
# Select relevant features and target
X = df[['Age', 'EstimatedSalary']].values
y = df['Purchased'].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# z score (-3 to +3)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train
```

```
array([[-0.06675246, -0.38634438],
       [ 0.79753468, -1.22993871],
       [ 0.11069205,  1.853544 ],
       [ 0.60129393, -0.90995465],
       [ 1.87685881, -1.28811763],
       [-0.57615058,  1.44629156],
       [ 0.3069328 , -0.53179168],
       [ 0.99377543,  0.10817643],
       [-1.16487283,  0.45724994],
       [-1.55735433,  0.31180264],
       [ 1.0918958 ,  0.45724994],
       [-0.18366908, -0.47361276],
       [ 0.20881242, -0.32816546],
       [ 0.3069328 ,  0.28271318],
       [-1.16487283, -1.57901222],
       [ 0.11069205,  0.25362372],
       [ 2.07309956,  1.73718616],
       [ 0.40505317, -0.18271817],
       [ 1.4843773 ,  2.11534913],
       [-0.37990983,  1.21357589],
```

```
[ 1.87685881,  1.50447048],  
[ 0.11069205,  0.02090805],  
[ 0.89565505, -1.31720709],  
[-1.36111358, -1.49174384],  
[-0.18366908, -0.5899706 ],  
[-0.57615058,  2.31897535],  
[ 0.99377543, -1.20084925],  
[-0.77239133,  1.06812859],  
[ 2.17121993, -1.05540195],  
[ 0.01257167, -0.26998655],  
[-0.4780302 , -1.14267033],  
[ 0.11069205,  0.02090805],  
[ 1.68061805, -0.90995465],  
[ 1.0918958 , -0.90995465],  
[ 0.50317355,  1.21357589],  
[-1.06675246, -1.46265438],  
[-1.16487283, -1.54992276],  
[-0.0855487 ,  0.66087616],  
[ 0.40505317, -0.47361276],  
[-0.28178945, -0.26998655],  
[-0.87051171,  0.13726589],  
[ 0.01257167,  0.28271318],  
[ 0.6994143 , -1.28811763],  
[ 1.58249768,  1.09721805],  
[ 0.79753468, -1.37538601],  
[-1.45923396, -1.22993871],  
[-0.0855487 ,  0.13726589],  
[ 0.40505317, -0.15362871],  
[-0.28178945,  0.02090805],  
[ 1.28813655,  2.20261751],  
[ 0.11069205,  0.74814454],  
[-1.36111358,  0.54451832],  
[ 1.97497918,  0.71905508],  
[-1.26299321, -1.40447546],  
[ 0.3069328 , -0.32816546],  
[-0.96863208,  0.54451832],  
[ 0.40505317,  0.28271318],  
[ 0.40505317,  1.09721805],
```

`X_test`

```
array([[ 0.79753468, -1.40447546],  
       [ 2.07309956,  0.51542886],  
       [-0.96863208, -0.76450736],  
       [ 0.99377543,  0.74814454],  
       [-0.87051171, -1.22993871],  
       [-0.77239133, -0.24089709],  
       [ 0.89565505,  1.06812859],  
       [-0.87051171,  0.36998156],  
       [ 0.20881242,  0.13726589],  
       [ 0.40505317, -0.15362871],  
       [-0.28178945, -0.15362871],  
       [ 1.4843773 , -1.05540195],  
       [-1.45923396, -0.64814952],  
       [-1.75359508, -1.37538601],  
       [-0.77239133,  0.4863394 ],  
       [-0.28178945,  1.09721805],  
       [ 1.38625693, -0.93904411],  
       [ 0.79753468,  0.10817643],  
       [ 0.11069205, -0.82268628],  
       [ 1.77873843, -0.29907601],  
       [-1.55735433, -1.25902817],  
       [-0.87051171,  0.28271318],  
       [ 0.89565505, -1.37538601],  
       [ 2.07309956,  0.16635535],  
       [-1.85171546, -1.49174384],  
       [ 1.28813655, -1.37538601],  
       [ 0.40505317,  0.28271318],  
       [-0.0855487 , -0.50270222],  
       [ 1.68061805,  1.59173886],  
       [-1.85171546, -1.43356492],  
       [ 0.79753468, -0.85177573],  
       [-1.85171546, -0.00818141],  
       [-0.18366908,  2.14443859],  
       [-0.96863208,  0.25362372],  
       [ 0.20881242,  1.06812859],  
       [-0.28178945,  0.13726589],  
       [-0.0855487 , -0.4445233 ],  
       [ 0.01257167, -0.15362871],  
       [-1.16487283, -1.17175979],  
       [-1.94983583, -0.06636033],  
       [ 0.99377543, -1.08449141],  
       [-1.36111358, -0.4445233 ],  
       [-1.94983583, -0.53179168],  
       [ 0.89565505, -1.46265438],  
       [-1.75359508, -0.61906006],  
       [ 0.60129393,  1.99899129],  
       [-0.87051171, -0.26998655],  
       [-0.67427095,  0.02090805],  
       [ 0.99377543, -0.85177573],
```

```
[ -0.37990983, -0.79359682],
[ -1.26299321,  0.25362372],
[ 1.4843773 ,  0.3408921 ],
[ 0.01257167, -0.4445233 ],
[ -1.26299321,  0.28271318],
[ -0.0855487 ,  0.28271318],
[ -1.06675246, -1.14267033],
[ 2.17121993,  0.92268129],
[ -1.16487283,  1.38811264],
```

```
model = Perceptron(max_iter=1000, eta0=1.0, random_state=42) # eta 0.0001 and 1.
model.fit(X_train, y_train)
```

▼ Perceptron ⓘ ⓘ
► Parameters

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f" Accuracy: {accuracy:.2f}")
```

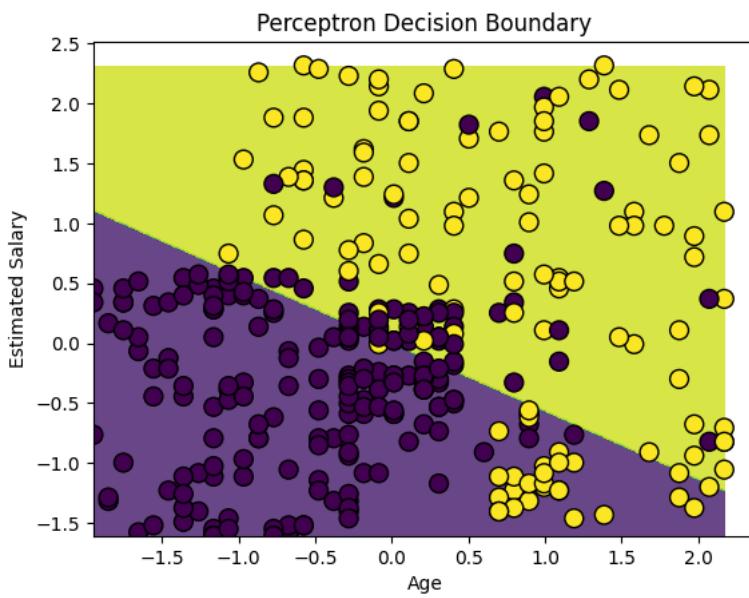
Accuracy: 0.76

```
# Assign columns to descriptive variable names
# X = df[['Age', 'EstimatedSalary']].values
age = X_train[:, 0]
salary = X_train[:, 1]
print (age)
```

```
[ -0.06675246  0.79753468  0.11069205  0.60129393  1.87685881 -0.57615058
 0.3069328  0.99377543 -1.16487283 -1.55735433  1.0918958 -0.18366908
 0.20881242 0.3069328 -1.16487283  0.11069205  2.07309956  0.40505317
1.4843773 -0.37990983  1.87685881  0.11069205  0.89565505 -1.36111358
-0.18366908 -0.57615058  0.99377543 -0.77239133  2.17121993  0.01257167
-0.4780302  0.11069205  1.68061805  1.0918958  0.50317355 -1.06675246
-1.16487283 -0.0855487  0.40505317 -0.28178945 -0.87051171  0.01257167
 0.6994143  1.58249768  0.79753468 -1.45923396 -0.0855487  0.40505317
-0.28178945  1.28813655  0.11069205 -1.36111358  1.97497918 -1.26299321
 0.3069328 -0.96863208  0.40505317  0.40505317  0.79753468  0.89565505
-0.4780302 -1.85171546  1.0918958 -0.67427095 -0.77239133  0.99377543
-0.57615058 -0.0855487 -1.94983583  0.40505317 -1.06675246  0.20881242
 1.87685881 -1.16487283 -0.87051171 -0.4780302  0.11069205
 1.58249768 -0.18366908  0.79753468  0.20881242 -0.0855487 -0.87051171
-0.0855487 -0.28178945  0.40505317 -0.28178945  0.3069328  0.20881242
-1.16487283 -0.67427095 -0.28178945 -0.28178945 -0.77239133  0.3069328
-0.96863208  0.79753468 -1.55735433  0.6994143 -0.77239133 -0.37990983
 0.20881242 -0.57615058 -1.45923396 -1.16487283  0.99377543  2.07309956
-0.28178945 -0.37990983  0.11069205 -1.06675246  2.17121993 -1.36111358
 1.97497918 1.4843773 -0.28178945 -0.18366908  1.87685881 -0.0855487
-0.4780302  0.3069328  1.38625693  1.38625693 -0.0855487 -1.16487283
-1.16487283 -1.16487283  0.3069328 -0.67427095 -0.0855487  0.01257167
-0.67427095 0.40505317  0.79753468 -0.18366908  0.01257167  1.0918958
 1.87685881 0.99377543  0.89565505 -0.4780302 -0.0855487 -1.75359508
 0.20881242 -0.28178945 -1.75359508  0.6994143 -1.16487283  1.97497918
 0.20881242 0.01257167 -0.0855487  1.0918958 -0.18366908  0.3069328
-0.28178945 0.50317355  0.3069328  0.01257167  1.97497918 -1.16487283
-0.57615058 0.3069328 -0.67427095 -0.28178945  0.99377543 -1.06675246
-0.28178945 0.40505317 -1.65547471  0.3069328 -0.4780302  0.11069205
-0.96863208 0.79753468 -1.36111358 -0.67427095  1.19001618 -1.75359508
-0.57615058 -1.75359508 -0.28178945 -0.87051171 -1.65547471 -0.28178945
 0.6994143 2.17121993 -0.28178945 -0.28178945 -1.94983583  0.40505317
 0.6994143 0.20881242 -1.36111358 -0.4780302  0.40505317  1.68061805
 1.19001618 0.99377543 -1.85171546  0.3069328  0.20881242 -1.06675246
-0.18366908 -0.28178945 -0.28178945 -0.37990983 -0.28178945  2.17121993
-1.26299321 -1.06675246 -1.65547471 -0.0855487 -0.28178945  1.58249768
 0.89565505 0.01257167 -0.28178945 -0.28178945  0.3069328 -0.0855487
 2.17121993 -0.96863208 -1.45923396 -1.45923396 -0.77239133 -1.26299321
 2.07309956 1.97497918 -1.94983583  0.79753468 -1.06675246  1.97497918
 0.99377543 0.0118958 1.19001618 1.4843773  0.20881242  1.38625693
 2.07309956 -0.18366908 0.40505317 -0.0855487  0.99377543  2.07309956
-0.0855487 0.20881242 0.0918958 -1.26299321 -0.77239133  0.20881242
-1.36111358 0.01257167 0.6994143 -1.65547471 -0.28178945 -0.28178945
-0.67427095 -0.77239133 0.89565505 -1.06675246 -0.28178945 -0.77239133
 0.11069205 -0.87051171 0.3069328 0.99377543 0.01257167 -0.57615058
 0.11069205 -1.16487283 0.11069205 -1.55735433 -0.18366908 2.17121993
-0.96863208 1.28813655 0.0918958 -0.18366908 1.77873843 -1.06675246
-1.36111358 -0.37990983 -0.96863208 1.19001618 -0.4780302 -0.87051171
-1.55735433 -0.77239133 0.89565505 0.79753468 0.99377543 0.89565505
 1.0918958 -0.57615058 0.01257167 -1.85171546 -0.0855487 0.89565505
 0.20881242 -0.18366908 -1.75359508 -0.67427095 0.3069328 0.40505317
-0.96863208 0.11069205 -0.96863208 0.01257167 0.89565505 -0.0855487
```

```
1.0918958  0.6994143 -0.28178945 -1.36111358 -1.16487283  0.50317355  
0.11069205 -0.57615058]
```

```
# Create a mesh grid using exact min and max values  
x_min, x_max = age.min(), age.max()  
y_min, y_max = salary.min(), salary.max()  
xx, yy = np.meshgrid(  
    np.arange(x_min, x_max, 0.01),  
    np.arange(y_min, y_max, 0.01))  
  
# Predict class labels for all points in the mesh  
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
  
# Plot the decision boundary  
plt.contourf(xx, yy, Z, alpha=0.8)  
plt.scatter(age, salary, c=y_train, edgecolors='k', marker='o', s=100)  
plt.title('Perceptron Decision Boundary')  
plt.xlabel('Age')  
plt.ylabel('Estimated Salary')  
plt.show()
```



Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.