Prirodoslovno-matematički fakultet Sveučilišta u Zagrebu

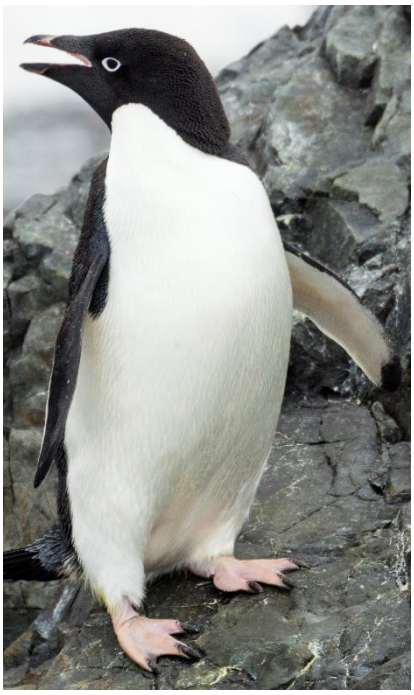
TRENIRANJE KVANTNOG MODELA STROJNOG UČENJA NA SKUPU PODATAKA PALMER ARCHIPELAGO (ANTARCTICA)

Zagreb, 14.02.2024. Petra Salaj

Sadržaj

1. UVOD

U ovome radu objasnit ću kako se trenira kvantni model strojnog učenja za rješavanje problema klasifikacije na stvarnom skupu podataka. Odabrala sam skup podataka o pingvinima koji su prikupili i učinili dostupnima dr. Kristen Gorman i Palmer Station, Antarctica LTER. (https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data/data) Znanstvenici su prikupili podatke o tri vrste pingvina; Adelie, Chinstrap i Gentoo, ali u ovom radu fokusirat ću se na vrstu Adelie. Pingvin Adelie najmanja je i najrasprostranjenija vrsta pingvina u Južnom oceanu te je jedna od dvije vrste koja se nalazi na Antarktici. Također, adelie pingvini jedna su od najlakših vrsta pingvina s plavo-crnim leđima i potpuno bijelim prsima i trbuhom. Imaju crnu glavu i kljun s prepoznatljivim bijelim prstenom oko svakog oka. (https://scr.ekolss.com/adelie-penguin-information) Mužjak i ženka pingvina ne razlikuju se mnogo kada je u pitanju izgled, ali razlikuju se u veličinama svojih obilježja. Dakle, problem klasifikacije kojim ćemo se baviti u ovom radu je određivanje spola pingvina Adelie na temelju 4 značajki; duljini i dubini kljuna u milimetrima (Culmen Length and Depth), duljini peraje u milimetrima (Flipper Length) i tjelesnoj težini u gramima. Seminar i kod su temeljeni na tutorialu „Training a Quantum Model on a Real Dataset“ (https://qiskit-community.github.io/qiskit-machine-learning/tutorials/02a_training_a_quantum_model_on_a_real_dataset.html#).



Slika 1: Pingvin Adelie (https://en.wikipedia.org/wiki/Ad%C3%A9lie_penguin#/media/File:Hope_Bay-2016-Trinity_Peninsula%E2%80%93Ad%C3%A9lie_penguin_(Pygoscelis_adeliae)_04.jpg)

Slika 2: Duljina i dubina kljuna (https://twitter.com/allison_horst/status/1270046411002753025)

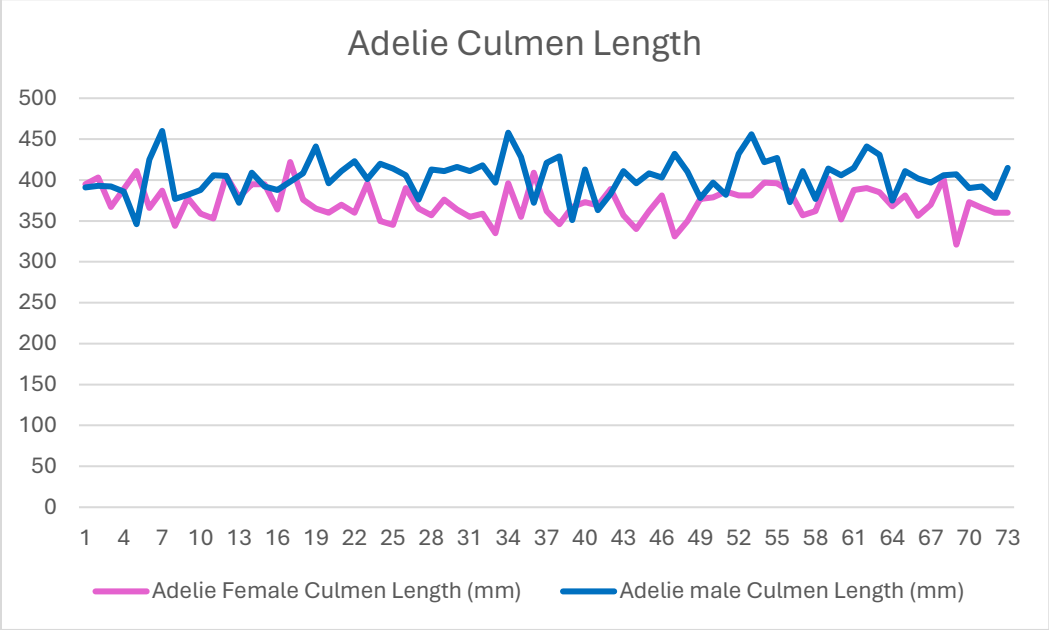
2. STROJNO UČENJE

Strojno učenje sastoji se od 7 osnovnih koraka:

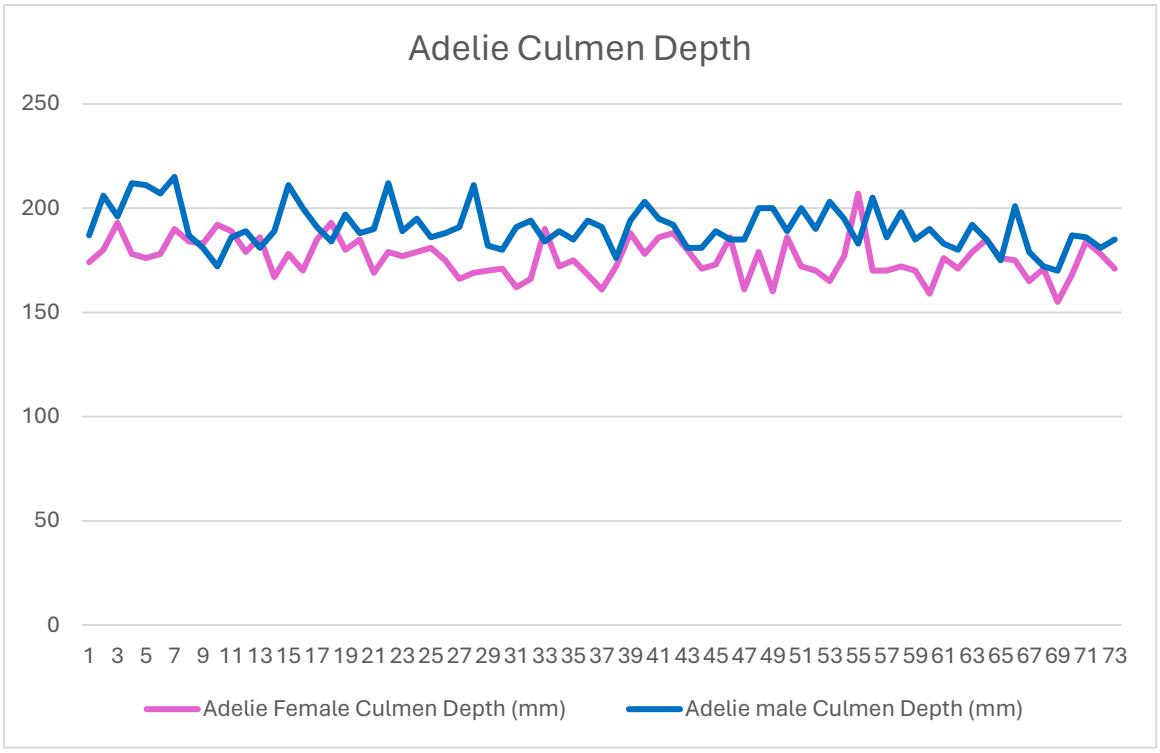
1. Prikupljanje podataka
2. Pripremanje podataka
3. Biranje modela
4. Treniranje modela
5. Ocjenjivanje modela
7. Podešavanje parametara

2.1. Prikupljanje i pripremanje podataka

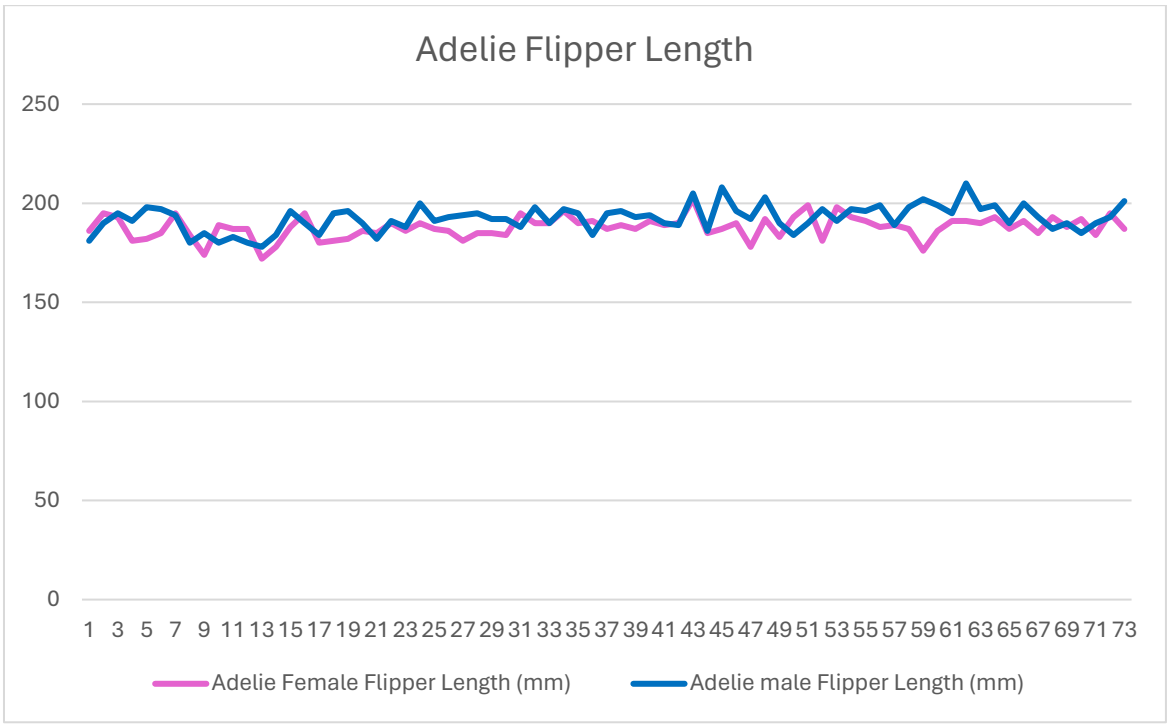
Skup podataka o pingvinima Adelie preuzela sam sa EDI Data Portala (https://portal.edirepository.org/nis/mapbrowse?packageid=knb-lter-pal.219.5). Ukupno ima 152 podataka o jedinkama pingvina vrste Adelie; 73 mužjaka, 73 ženke i 6 bez određenog spola. Prikupljeni su podaci o brojnim značajkama poput regije, otoku, stadiju razvija i tjelesnim obilježjima. U ovom seminaru odlučila sam se fokusirati samo na 4 značajke spomenute u prethodnom odjeljku zbog čega sam podatke o ostalima maknula iz tablice. Također sam iz daljnje analize izostavila podatke o 6 pingvina kojima nije određen spol. Napravila sam usporedbe obilježja kod mužjaka i ženki najprije na originalnim podacima u Excelu, a kasnije i u Pythonu nakon normalizacije značajki. Iz grafova 1, 2, 3 i 4 vidimo da se kod duljine i dubine kljuna te duljini peraje vrijednosti dosta isprepliću, jasnija razlika je vidljiva jedino kod tjelesne težine gdje mužjaci imaju malo veću masu od ženaka. Također isti zaključak donosimo iz grafova s normaliziranim značajkama sa Slike 3.



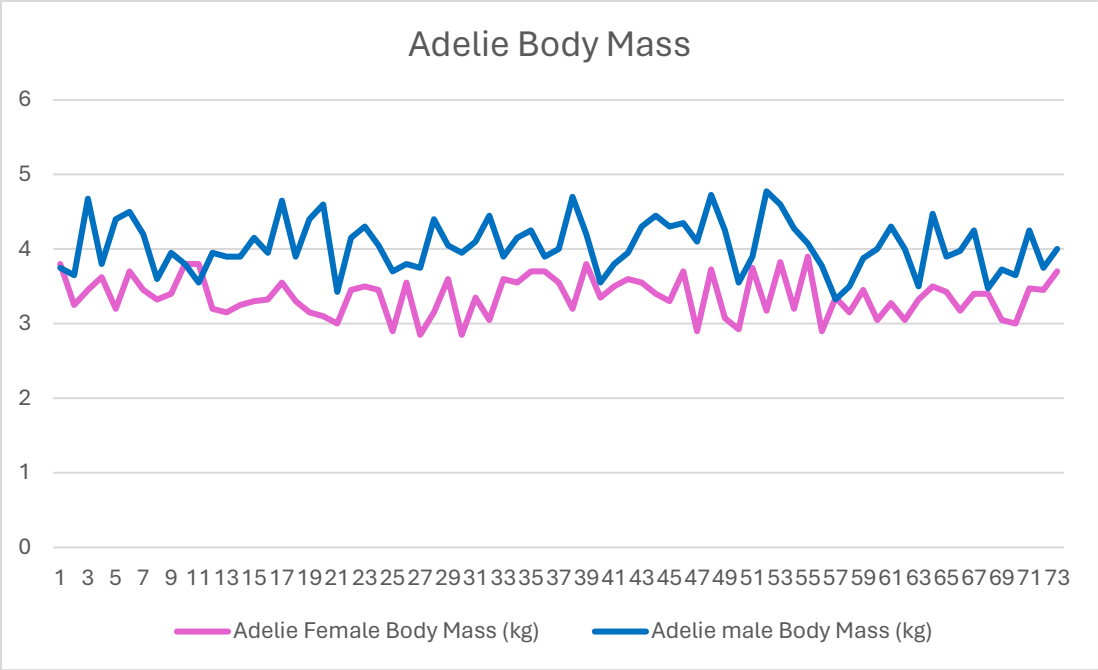
Graf 1: Duljine kljuna kod mužjaka i ženki u mm



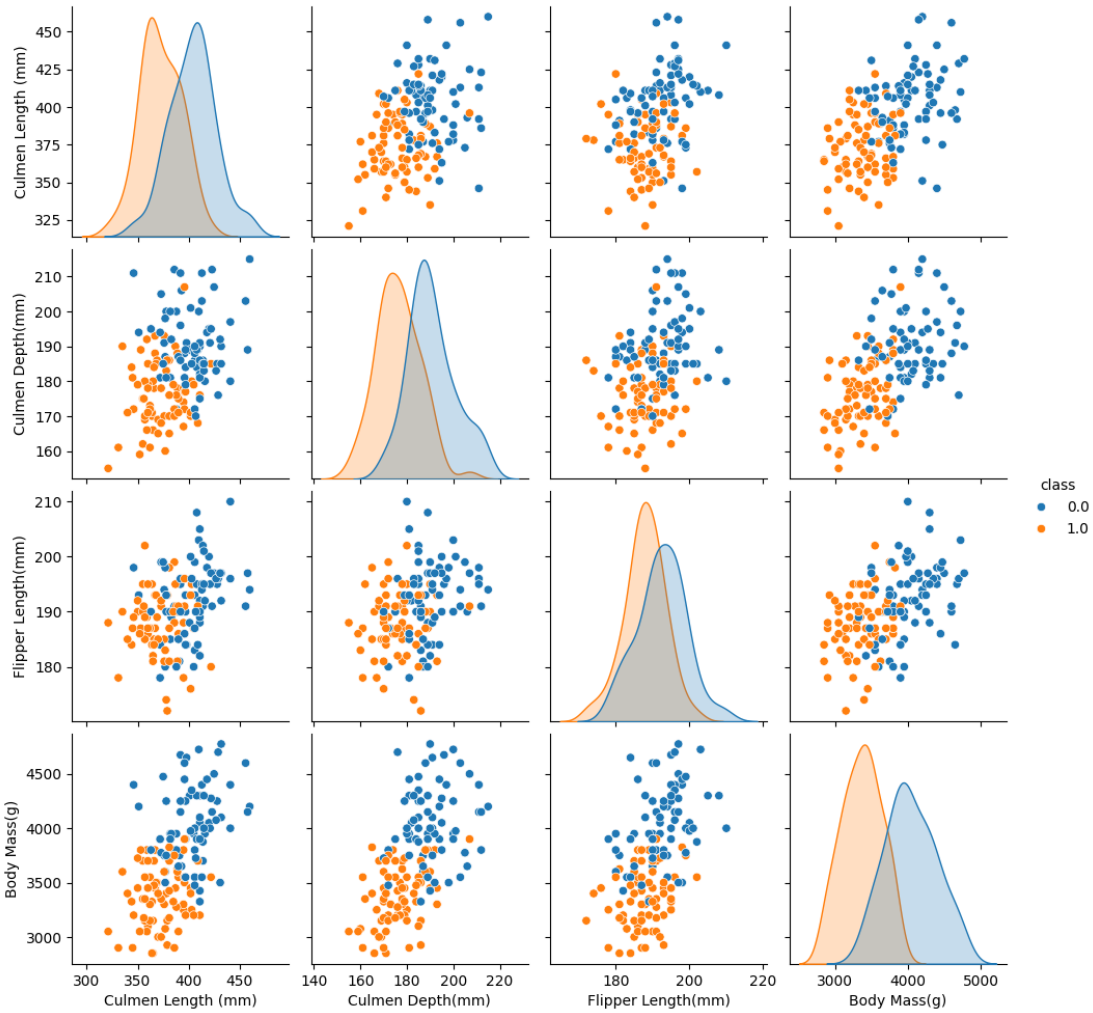
Graf 2: Dubine kljuna mužjaka i ženki u mm



Graf 3: Duljine peraje kod mužjaka i ženki u mm



Graf 4: Tjelesna težina kod mužjaka i ženki u kg



Slika 3: Usporedbe normaliziranih značajki kod mužjaka (klasa 0.0) i ženki (klasa 1.0)

Normalizacija značajki česta je tehnika u strojnom učenju koja često dovodi do bolje numeričke stabilnosti i konvergencije algoritma. Naime, primjenjujemo jednostavnu transformaciju na podatke kako bi ih sve prikazali na istoj skali. U našem slučaju, želimo da nam svi podaci imaju vrijednosti iz segmenta [0, 1]. To postižemo tako da za svaku značajku pronađemo minimalnu (MIN) i maksimalnu vrijednost (MAX) i skaliranu vrijednost dobijemo po formuli $scaledValue = \frac{originalValue - MIN}{(MAX+1) - MIN}$. Upravo to radi `MinMaxScaler` iz Python biblioteke scikit-learn na Slici 4.

```
18  features=[]
19  labels=np.array([])
20
21  with open(r'C:\Users\Petra\OneDrive\Radna površina\kvantno_projekt\adelie.csv', newline='') as csvfile:
22      adelie = csv.DictReader(csvfile)
23      for pingvin in adelie:
24          pom_array = [int(pingvin['Culmen_Length(mm)']),int(pingvin['Culmen_Depth(mm)']),int(pingvin['Flipper_Length(mm)']),int(pingvin['Body_Mass(g)'])]
25          features.append(pom_array)
26          labels=np.append(labels,int(pingvin['Sex']))
27
28  print("Features: ",features)
29  print("Labels: ",labels)
30  featureNames=['Culmen Length (mm)','Culmen Depth(mm)','Flipper Length(mm)','Body Mass(g)']
31  normalizedFeatures = MinMaxScaler().fit_transform(features)
32  print("Normalized Features: ",normalizedFeatures)
```

Slika 4: Dio koda za inicijalizaciju polja sa vrijednostima značajka (array features) i normalizaciju podataka

```
Features:  [[391, 187, 181, 3750], [395, 174, 186, 3800], [403, 180, 195, 3250], [367, 193, 193, 3450], [393, 206, 190, 3650], [389, 178, 181, 3625], [392, 196, 195, 4675], [411, 176,
182, 3200], [386, 212, 191, 3800], [346, 211, 198, 4400], [366, 178, 185, 3700], [387, 190, 195, 3450], [425, 207, 197, 4500], [344, 184, 184, 3325], [460, 215, 194, 4200], [378, 183,
174, 3400], [377, 187, 180, 3600], [359, 192, 189, 3800], [382, 181, 185, 3950], [388, 172, 180, 3800], [353, 189, 187, 3800], [406, 186, 183, 3550], [405, 179, 187, 3200], [379, 186,
172, 3150], [405, 189, 180, 3950], [395, 167, 178, 3250], [372, 181, 178, 3900], [395, 178, 188, 3300], [409, 189, 184, 3900], [364, 170, 195, 3325], [392, 211, 196, 4150], [388, 200,
190, 3950], [422, 185, 180, 3550], [376, 193, 181, 3300], [398, 191, 184, 4650], [365, 180, 182, 3150], [408, 184, 195, 3900], [360, 185, 186, 3100], [441, 197, 196, 4400], [370, 169,
185, 3000], [396, 188, 190, 4600], [411, 190, 182, 3425], [360, 179, 190, 3450], [423, 212, 191, 4150], [396, 177, 186, 3500], [401, 189, 188, 4300], [350, 179, 190, 3450], [420, 195,
200, 4050], [345, 181, 187, 2900], [414, 186, 191, 3700], [390, 175, 186, 3550], [406, 188, 193, 3800], [365, 166, 181, 2850], [376, 191, 194, 3750], [357, 169, 185, 3150], [413, 211,
195, 4400], [376, 170, 185, 3600], [411, 182, 192, 4050], [364, 171, 184, 2850], [416, 180, 192, 3950], [355, 162, 195, 3350], [411, 191, 188, 4100], [359, 166, 190, 3050], [418, 194,
198, 4450], [335, 190, 190, 3600], [397, 184, 190, 3900], [396, 172, 196, 3550], [458, 189, 197, 4150], [355, 175, 190, 3700], [428, 185, 195, 4250], [409, 168, 191, 3700], [372, 194,
184, 3900], [362, 161, 187, 3550], [421, 191, 195, 4000], [346, 172, 189, 3200], [429, 176, 196, 4700], [367, 188, 187, 3800], [351, 194, 193, 4200], [373, 178, 191, 3350], [413, 203,
194, 3550], [363, 195, 190, 3800], [369, 186, 189, 3500], [383, 192, 189, 3950], [389, 188, 190, 3600], [357, 180, 202, 3550], [411, 181, 205, 4300], [340, 171, 185, 3400], [396, 181,
186, 4450], [362, 173, 187, 3300], [408, 189, 208, 4300], [381, 186, 190, 3700], [403, 185, 196, 4350], [331, 161, 178, 2900], [432, 185, 192, 4100], [350, 179, 192, 3725], [410, 200,
203, 4725], [377, 160, 183, 3075], [378, 200, 190, 4250], [379, 186, 193, 2925], [397, 189, 184, 3550], [386, 172, 199, 3750], [382, 200, 190, 3900], [381, 170, 181, 3175], [432, 190,
197, 4775], [381, 165, 198, 3825], [456, 203, 191, 4600], [397, 177, 193, 3200], [422, 195, 197, 4275], [396, 207, 191, 3900], [427, 183, 196, 4075], [386, 170, 188, 2900], [373, 205,
199, 3775], [357, 170, 189, 3350], [411, 186, 189, 3325], [362, 172, 187, 3150], [377, 198, 198, 3500], [402, 170, 176, 3450], [414, 185, 202, 3875], [352, 159, 186, 3050], [406, 190,
199, 4000], [388, 176, 191, 3275], [415, 183, 195, 4300], [390, 171, 191, 3050], [441, 180, 210, 4000], [385, 179, 190, 3325], [431, 192, 197, 3500], [368, 185, 193, 3500], [375, 185,
199, 4475], [381, 176, 187, 3425], [411, 175, 190, 3900], [356, 175, 191, 3175], [402, 201, 200, 3975], [370, 165, 185, 3400], [397, 179, 193, 4250], [402, 171, 193, 3400], [406, 172,
187, 3475], [321, 155, 188, 3050], [407, 170, 190, 3725], [373, 168, 192, 3000], [390, 187, 185, 3650], [392, 186, 190, 4250], [366, 184, 184, 3475], [360, 178, 195, 3450], [378, 181,
193, 3750], [360, 171, 187, 3700], [415, 185, 201, 4000]]
```

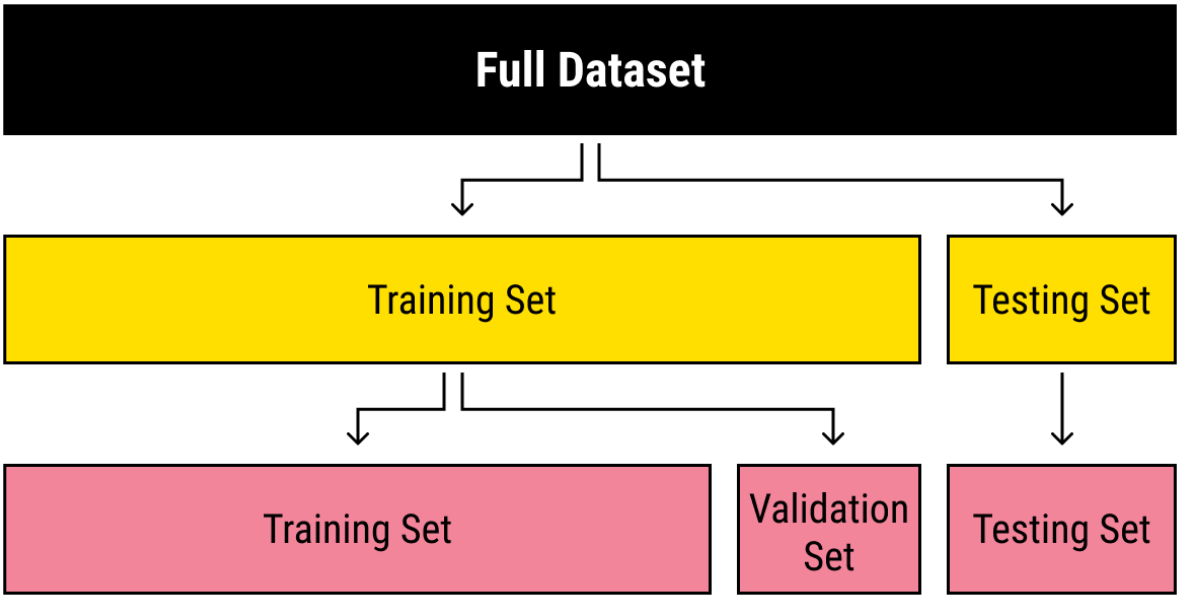
Slika 5: Ispis originalnih vrijednosti značajka (rezultat linije 28)

```
Normalized Features:  [[0.50359712 0.53333333 0.23684211 0.46753247]
 [0.5323741  0.31666667 0.36842105 0.49350649]
 [0.58992806 0.41666667 0.60526316 0.20779221]
 [0.33093525 0.63333333 0.55263158 0.31168831]
 [0.51798561 0.85       0.47368421 0.41558442]
 [0.48920863 0.38333333 0.23684211 0.4025974 ]
 [0.51079137 0.68333333 0.60526316 0.94805195]
 [0.64748201 0.35       0.26315789 0.18181818]
 [0.4676259  0.95       0.5        0.49350649]
 [0.17985612 0.93333333 0.68421053 0.80519481]
 [0.32374101 0.38333333 0.34210526 0.44155844]
 [0.47482014 0.58333333 0.60526316 0.31168831]
 [0.74820144 0.86666667 0.65789474 0.85714286]
 [0.16546763 0.48333333 0.31578947 0.24675325]
 [1.         1.         0.57894737 0.7012987 ]
```

Slika 6: Ispis normaliziranih vrijednosti značajka (rezultat linije 32)

2.2. Biranje, treniranje i ocjenjivanje modela

Prije nego što počnemo s treniranjem modela, treba podijeliti skup podataka u dva dijela: skup podataka za treniranje i testni skup podataka. Prvi ćemo koristiti za treniranje modela, a drugi za provjeru uspješnosti naših modela na modelu nevidljivim podacima.



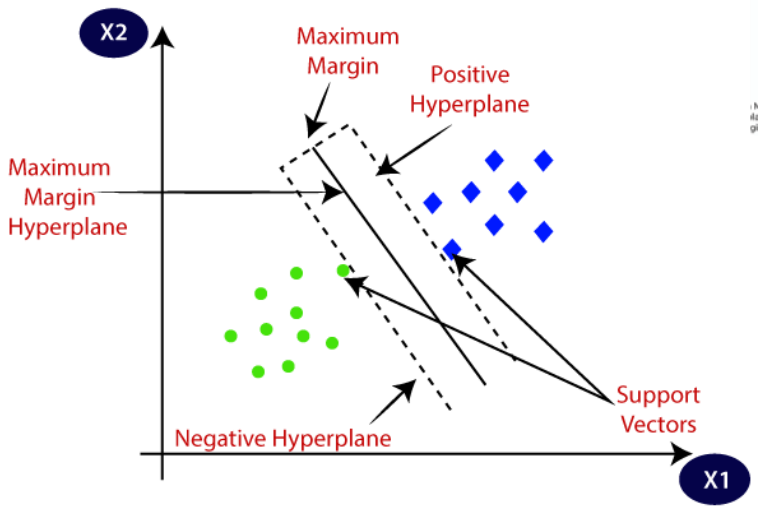
Slika 7: Podjela skupa podataka na skup podataka za treniranje i testni skup podataka (https://labelyourdata.com/articles/what-is-dataset-in-machine-learning)

```
40  algorithm_globals.random_seed = 123
41  train_features, test_features, train_labels, test_labels = train_test_split(
42      normalizedFeatures, labels, train_size=0.8, random_state=algorithm_globals.random_seed
43  )
```

Slika 8: Dio koda za podjelu skupa podataka

Na Slici 8 vidimo da podjelu umjesto nas radi funcija train_test_split() iz Python biblioteke scikit-learn za skup podataka za treniranje uzima 80% podataka iz početnog skupa podataka.

2.2.1. Klasični model



Slika 9: SVM model za klasifikaciju 2 klase u 2D prostoru (https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm)

Support Vector Machine (SVM) je algoritam strojnog učenja koji se koristi za probleme klasifikacije. Glavni cilj SVM algoritma je pronaći optimalnu hiperravninu (Maximumum Margin Hyperplane sa Slike 9) u N-dimenzionalnom prostoru koja može razdvojiti podatkovne točke u različitim klasama u prostoru značajki. Hiperravnina nastoji da razlika između najbližih točaka različitih klasa bude što je moguće veća. Dimenzija hiperravnine ovisi o broju značajki. Dakle, ako je broj ulaznih značajki dva, tada je hiperravnina samo linija. U našem slučaju imamo 4 značajke pa je dimenzija hiperravnine jednaka 4. (https://www.geeksforgeeks.org/support-vector-machine-algorithm/) Ovaj model koristimo za treniranje pomoću Support Vector Classifier iz biblioteke scikit-learn. Radi jednostavnosti, ne podešavamo nikakve parametre i oslanjamo se na zadane vrijednosti.

```
45  svc = SVC()
46  _ = svc.fit(train_features, train_labels)
47
48  train_score_c4 = svc.score(train_features, train_labels)
49  test_score_c4 = svc.score(test_features, test_labels)
50
51  print(f"Classical SVC on the training dataset: {train_score_c4:.2f}")
52  print(f"Classical SVC on the test dataset:     {test_score_c4:.2f}")
```

Slika 10: Dio koda SVM algoritma za treniranje klasičnog modela

```
Classical SVC on the training dataset: 0.93
Classical SVC on the test dataset:     0.93
```

Slika 11: Rezultati klasičnog modela

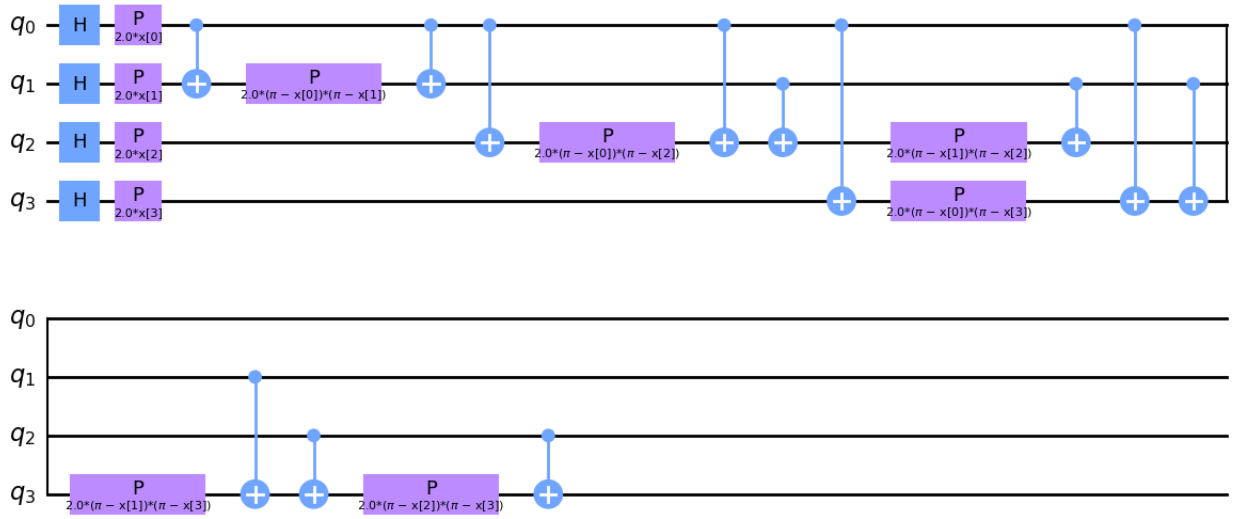
Rezultati sa Slike 11 sastoje se od 2 vrijednosti. Prva vrijednost predstavlja točnost modela na skupu za provjeru valjanosti koji je podskup skupa podataka za treniranje (Validation set sa Slike 7), a druga vrijednost predstavlja točnost modela na testnom skupu podataka. Skup za provjeru valjanosti koristi se tijekom faze treniranja modela za pružanje nepristrane procjene performansi modela i za fino podešavanje parametara modela. S druge strane, testni skup se koristi nakon što je model u potpunosti osposobljen za procjenu izvedbe modela na potpuno nevidljivim podacima. (https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data) Kao što vidimo na Slici 11 klasični model ima visoku točnost od 93% i na skupu za provjeru valjanosti i na testnom skupu.

2.2.2. Kvantni model

Za primjer kvantnog modela za problem klasifikacije, istrenirat ćemo Variational Quantum Classifier (VQC). VQC je najjednostavniji klasifikator dostupan u Qiskit Machine Learningu i dobra je polazna točka za početnike u kvantnom strojnom učenju. Dva središnja elementa klase VQC su mapa značajki i anzatz. Naši podaci su klasični, što znači da se sastoje od skupa bitova, a ne od qubita. Trebamo način da kodiramo podatke kao qubite. Ovaj proces je ključan ako želimo dobiti učinkovit kvantni model. Ovo preslikavanje (mapping) obično nazivamo kodiranjem podataka, ugrađivanjem podataka ili učitavanjem podataka i to je uloga mape značajki (feature_map na Slici 12). Naš odabir mape značajki bit će ZZFeatureMap. ZZFeatureMap jedna je od standardnih mapa značajki u Qiskit biblioteci sklopova. Prosljeđujemo num_features kao feature_dimension, što znači da će mapa značajki imati num_features ili 4 qubita. Na dijagramu mape značajki sa Slike 13 vidimo parametre x[0], ..., x[3] koji su rezervirana mjesta za naše značajke.

```
54  num_features = normalizedFeatures.shape[1]
55
56  feature_map = ZZFeatureMap(feature_dimension=num_features, reps=1)
57  feature_map.decompose().draw(output="mpl", style="clifford", fold=20)
58  plt.show()
```

Slika 12: Dio koda za mapu značajki

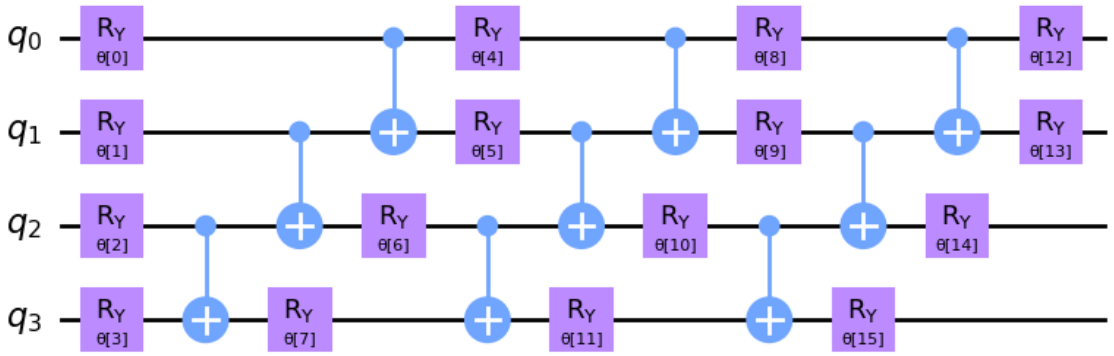


Slika 13: Dijagram mape značajki (rezultat koda sa Slike 12)

Nakon što se podaci učitaju, moramo odmah primijeniti parametrizirani kvantni krug. Ovaj sklop je izravna analogija slojeva u klasičnim neuronskim mrežama. Ima skup podesivih parametara ili težina. Težine su optimizirane tako da minimiziraju funkciju cilja (engl. objective function). Ova funkcija karakterizira udaljenost između predviđanja i poznatih označenih podataka. Parametrizirani kvantni krug također se naziva parametrizirano probno stanje, varijacijski oblik ili ansatz. Primijetimo na dijagramu sa Slike 15 ponavljajuću strukturu anzatz kruga. Broj ovih ponavljanja definiramo pomoću parametra reps.

```
60   ansatz = RealAmplitudes(num_qubits=num_features, reps=3)
61   ansatz.decompose().draw(output="mpl", style="clifford", fold=20)
62   plt.show()
```

Slika 14: Dio koda za ansatz



Slika 15: Dijagram ansatz-a (rezultat koda sa Slike 14)

Sklop sa Slike 15 ima 16 parametara nazvanih θ[0], ..., θ[15]. Ovo su težine klasifikatora koji treniramo.

```
64  optimizer = COBYLA(maxiter=100)
65
66  service = QiskitRuntimeService(channel="ibm_quantum",token="4e30e9f296b475eb6739cd26c0793
67  backend = service.get_backend("ibmq_qasm_simulator")
68
69  sampler = Sampler(backend=backend)
```

Slika 16: Dio koda za algoritam optimizacije COBYLA i pokretanje na simulatoru kvantnog računala

Na Slici 16 u liniji 64 biramo algoritam optimizacije koji ćemo koristiti u procesu treniranja modela. Ovaj je korak sličan onome što možete pronaći u klasičnim okvirima dubokog učenja. Kako bi proces treninga bio brži, odabiremo optimizator bez gradijenata. U sljedećim linijama koda sa Slike 16 definiramo gdje ćemo trenirati naš klasifikator. Možemo trenirati na simulatoru ili pravom kvantnom računalu. Ovdje ćemo koristiti simulator. Stvaramo instancu primitive Sampler. Ovo je referentna implementacija koja se temelji na vektoru stanja. Korištenjem qiskit runtime servisa možemo stvoriti sampler koji podržava kvantno računalo.

```
71  objective_func_vals = []
72  plt.rcParams["figure.figsize"] = (12, 6)
73
74
75  def callback_graph(weights, obj_func_eval):
76      clear_output(wait=True)
77      objective_func_vals.append(obj_func_eval)
78      plt.title("Objective function value against iteration")
79      plt.xlabel("Iteration")
80      plt.ylabel("Objective function value")
81      plt.plot(range(len(objective_func_vals)), objective_func_vals)
82      plt.show()
```

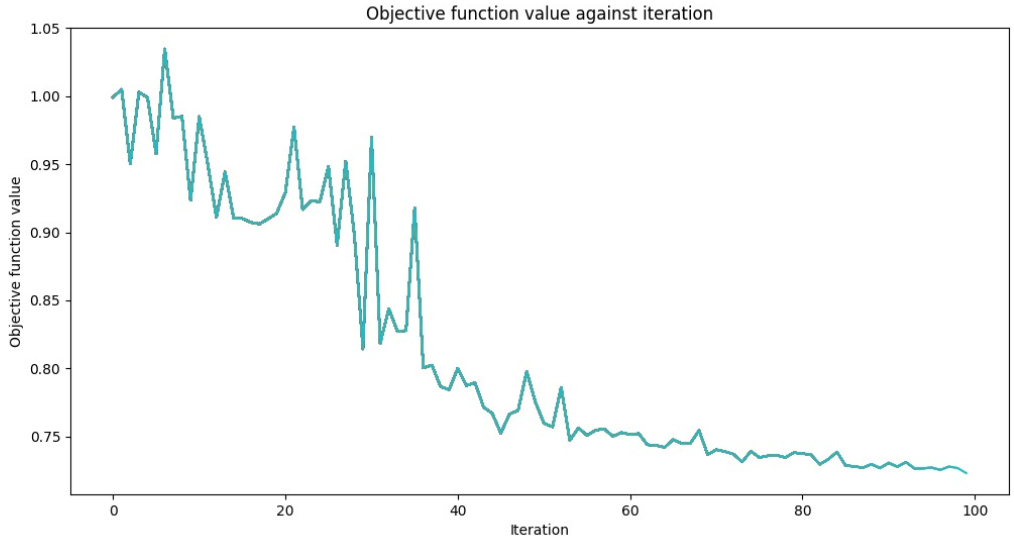
Slika 17: funkcija callback_graph

Na Slici 17 dodajemo funkciju povratnog poziva pod nazivom callback_graph. VQC će ovu funkciju pozvati za svaku procjenu funkcije cilja s dva parametra: trenutne težine i vrijednost funkcije cilja pri tim težinama. Naš povratni poziv će dodati vrijednost funkcije cilja nizu kako bismo mogli iscrtati iteraciju u odnosu na vrijednost funkcije cilja. Povratni poziv ažurirat će dijagram pri svakoj iteraciji. Maksimalan broj iteracija određujemo parametrom maxiter u liniji 64 na Slici 16. Najprije sam pokrenula kod na maksimalnom broju iteracija jednakom 100, a zatim i na 40. Konačni dijagram nakon 100 iteracija vidljiv je na Slici 19 na idućoj stranici.

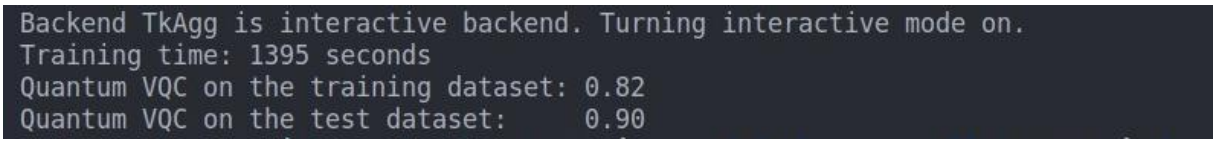
```
85  vqc = VQC(
86      sampler=sampler,
87      feature_map=feature_map,
88      ansatz=ansatz,
89      optimizer=optimizer,
90      callback=callback_graph,
91  )
92
93  # clear objective value history
94  objective_func_vals = []
95
96  start = time.time()
97  vqc.fit(train_features, train_labels)
98  elapsed = time.time() - start
99
100 print(f"Training time: {round(elapsed)} seconds")
101
102
103 train_score_q4 = vqc.score(train_features, train_labels)
104 test_score_q4 = vqc.score(test_features, test_labels)
105
106 print(f"Quantum VQC on the training dataset: {train_score_q4:.2f}"
107 print(f"Quantum VQC on the test dataset:     {test_score_q4:.2f}")
108
```

Slika 18: Kod za klasifikator VQC

Na Slici 18 konstruiramo klasifikator (linije 85-91) i prilagođavamo ga (linija 97). VQC ("variational quantum classifier") uzima mapu značajki i ansatz i automatski konstruira kvantnu neuronsku mrežu.



Slika 19: Rezultat poziva vqc.fit (linija koda 97) za 100 iteracija na 4 značajke

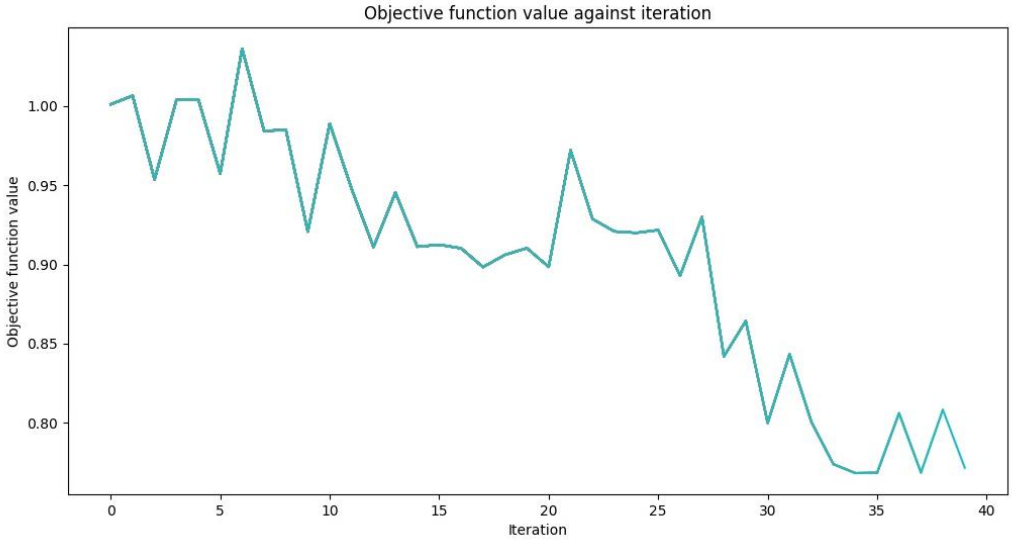


Slika 20: Rezultati VQC za 100 iteracija na 4 značajki

Treniranje kvantnog modela na simulatoru kvantnog računala je potrajalo malo više od 23 minute i dobili smo točnost modela na testnom skupu 90% što je dosta blizu točnosti klasičnog modela od 93%. Budući da imamo samo 10 minuta mjesečno na pravom kvantnom računala potrebno je podesiti parametre kako bi treniranje modela bilo moguće pokrenuti i na njemu.

2.3. Podešavanje parametara

Najprije sam smanjila maksimalni broj iteracija sa 100 na 40 i pokrenula prethodni kod na simulatoru. Na Slici 22 vidimo da se točnost kvantnog modela na testnom skupu smanjila na 73% (s prethodnih 90%). Također smanjenje broja iteracija smanjilo je trajanje treniranja modela na oko 10 minuta. Pokušala sam pokrenuti algoritam na pravom kvantnom računalu ibm_brisbane, međutim svaka iteracija trajala je malo više od 2 minute i za izvršenje svake iteracije trebalo je ponovno čekati u redu za daljnje pokretanje algoritma što je trajalo 15-35 minuta po iteraciji. Nakon samo 3 iteracije potrošila sam 7 min i 38 sekundi vremena, nakon čega je uslijedilo upozorenje sa Slike 23 da će 4.iteracija prekoračiti vrijeme od 10 minuta i izvršavanje biti prekinuto. Daljnja podešavanja parametara nastavila sam raditi na simulatoru.



Slika 21: Rezultat poziva vqc.fit (linija koda 97) za 40 iteracija funkcije callback_graph i 4 značajke

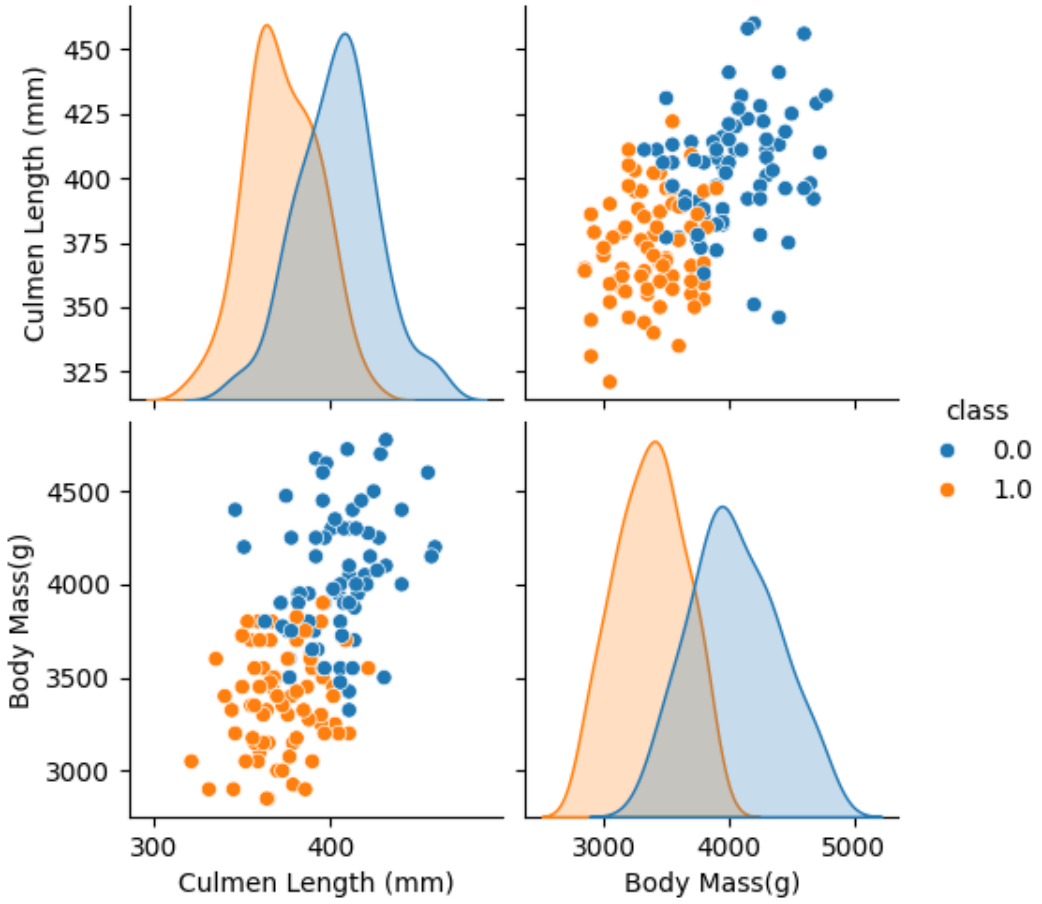
```
Training time: 620 seconds
Quantum VQC on the training dataset: 0.82
Quantum VQC on the test dataset:     0.73
```

Slika 22: Rezultati VQC za 40 iteracija na 4 značajki

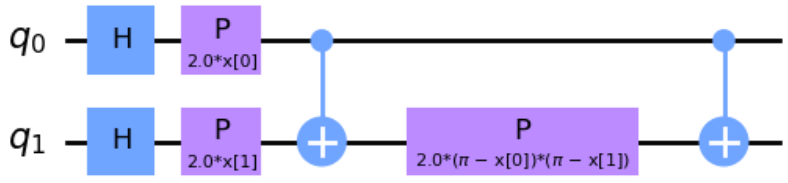
```
g: Your current pending jobs are estimated to consume 332.53742855000417 quantum seconds, but you only have 317 quantum seconds left in your mon
thly quota; therefore, it is likely this job will be canceled
  warnings.warn(warning_message)
```

Slika 23: Upozorenje o prekoračenju vremena izvršavanja od 10 minuta

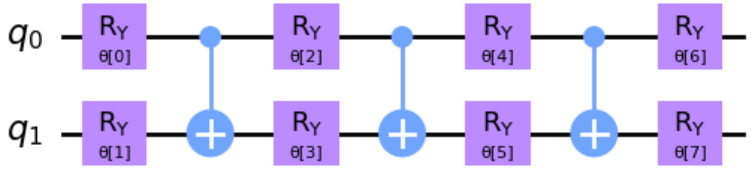
U idućem podešavanju parametara ostavila sam maksimalan broj iteracija 40, ali sam smanjila broj parametara na 2 u kojima se mužjaci i ženke najmanje podudaraju, a to su duljina kljuna u milimetrima i tjelesna težina u gramima. Slike 25 i 26 prikazuju dijagrame mape značajke i dijagram ansatz-a za 2 značajke.



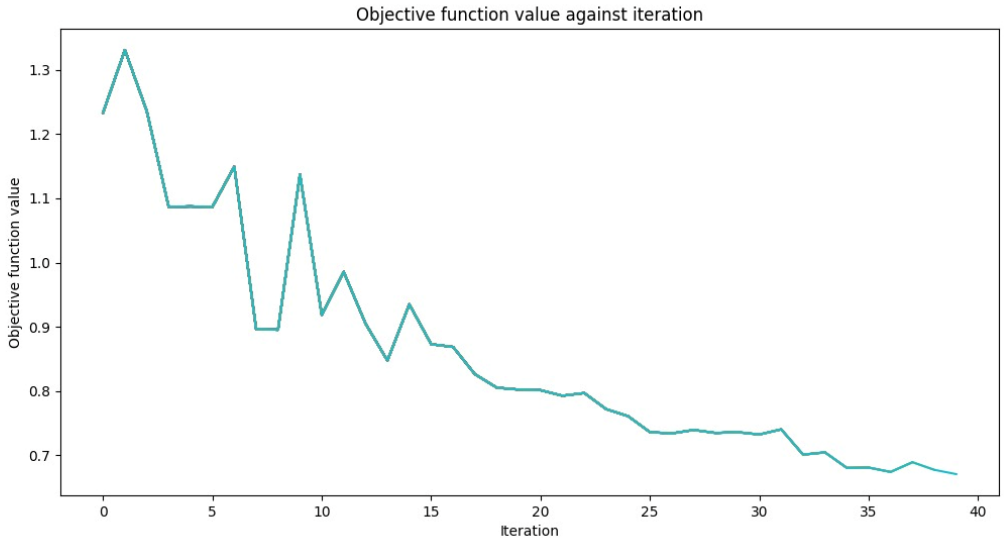
Slika 24: Usporedbe normaliziranih značajki kod mužjaka (klasa 0.0) i ženki (klasa 1.0)



Slika 25: Dijagram mape značajki



Slika 26: Dijagram ansatz-a



Slika 27: Rezultat poziva vqc.fit za 40 iteracija i 2 značajke

```
Classical SVC on the training dataset: 0.88
Classical SVC on the test dataset:     0.90
Training time: 465 seconds
Quantum VQC on the training dataset: 0.80
Quantum VQC on the test dataset:     0.87
```

Slika 28: Rezultati klasičnog i kvantnog VQC modela za 40 iteracija na 2 značajke

Smanjenjem broja značajki smanjila se točnost i klasičnog i kvantnog modela, ali se vrijeme treniranja kvantnog modela očekivano smanjilo na malo manje od 8 minuta.

3. ZAKLJUČAK

MODEL	NUMBER OF FEATURES	NUMBER OF ITERATIONS	VALIDATION SET SCORES	TEST SET SCORES	TIME OF TRAINING/s
SVC	4	-	0.93	0.93	-
VQC	4	100	0.82	0.90	1395
VQC	4	40	0.82	0.73	620
SVC	2	-	0.88	0.90	-
VQC	2	40	0.80	0.87	465

Tablica 1: Rezultati svih varijacija klasičnog i kvantnog modela

Klasični model strojnog učenja pokazuju bolje performanse od kvantnog modela. Međutim, vidimo da se za veliki broj iteracija kod kvantnog klasifikatora VQC rezultati na testnom skupu podataka ne razlikuju mnogo u točnosti. Klasični model SVM daje točnost od 93% dok kvantni daje točnost od 90%. Razlika je vidljiva na rezultatima za skup za provjeru valjanosti u kojem klasični model sa točnosti 93% daje bolje rezultate od kvantnog s točnosti 82%. Ali veliki nedostatak kvantog modela je dugo vrijeme izvršavanja od 23 minute dok za klasični model mjerenja vremena nije ni potrebno jer je gotovo u nekoliko sekundi. Očekivano za smanjeni broj iteracija točnost kvantnog modela na testnom skupu podataka dosta pada (za čak 17%). Također, za smanjeni broj značajki točnost oba modela pada (za 5% kod klasičnog i 2% kod kvantnog), ali i dalje je klasični model točniji od kvantnog modela i na skupu podataka za provjeru valjanosti i na testnom skupu podataka. Kvantni modeli su napredovali zadnjih godina, ali što se tiče točnosti i resursa (vrijeme i novac) klasični modeli su i dalje isplativiji.

4. LITERATURA

https://www.kaggle.com/datasets/parulpandey/palmer-archipelago-antarctica-penguin-data/data

https://qiskit-community.github.io/qiskit-machine-learning/tutorials/02a_training_a_quantum_model_on_a_real_dataset.html#

https://www.geeksforgeeks.org/support-vector-machine-algorithm/

https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data

https://en.wikipedia.org/wiki/Ad%C3%A9lie_penguin#/media/File:Hope_Bay-2016-Trinity_Peninsula%E2%80%93Ad%C3%A9lie_penguin_(Pygoscelis_adeliae)_04.jpg

(https://twitter.com/allison_horst/status/1270046411002753025

https://labelyourdata.com/articles/what-is-dataset-in-machine-learning

https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm