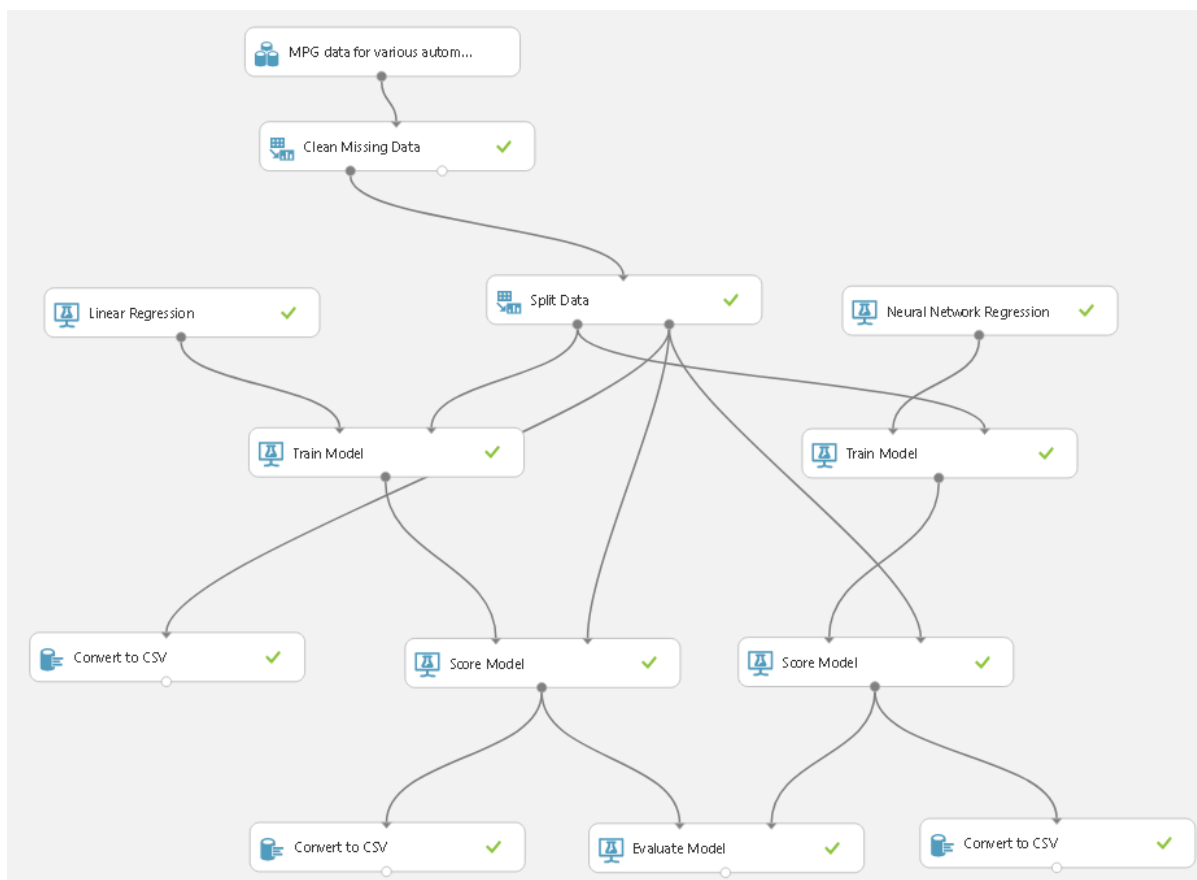


Zadatak 1.

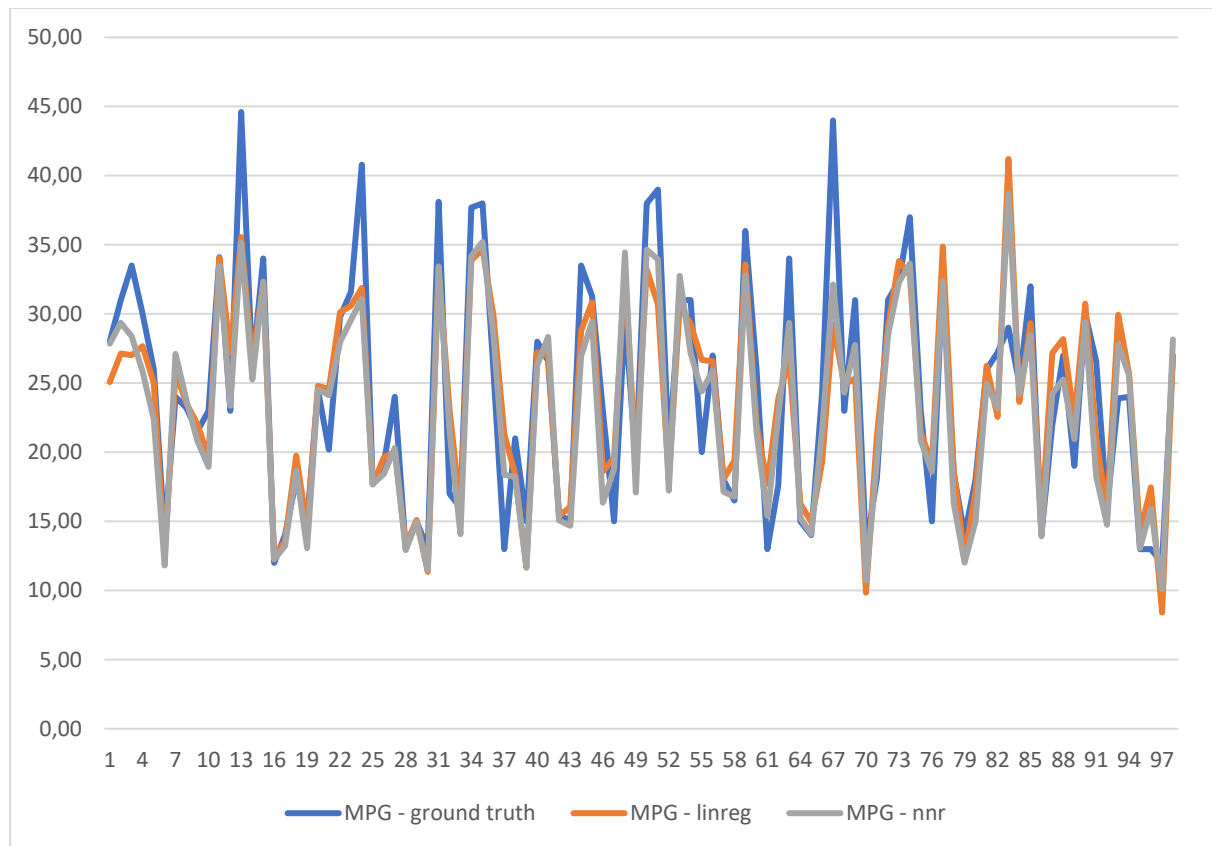
Moramo složiti eksperiment na Azure ML platformi.



Kao odabrane podatke smo koristili tablicu „MPG data for various automobiles“, gdje su traženi (target) podatci MPG, milja po galonu goriva. Prvo predajemo podatke kroz čišćenje u slučaju da u podacima postoje redovi sa elementima koji nedostaju. Nakon toga podatke rastavljamo u dva skupa, jedan za treniranje (75%) i drugi za testiranje podataka (25%). Podatke za treniranje šaljemo dalje za treniranje naša dva modela, dok podatke za testiranje šaljemo za konačno bodovanje i evaluaciju modela.

Dodali smo elemente „Convert to CSV“ da možemo izlučiti potrebne podatke za usporedbu metoda. Za grafičku usporedbu koristimo vrijednosti dobivene od split data za testiranje, to zajedno sa dobivenim vrijednostima od „Score Model“-a koristimo da se izradi graf.

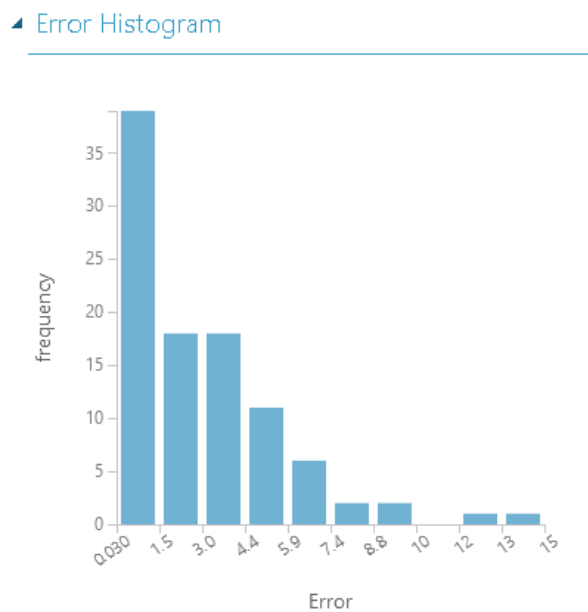
Dolje je prikazan graf sa pripadajućim vrijednostima. Vidimo da oba modela relativno uspješno prate originalni graf, negdje jedan i drugi griješe sa predviđenim vrijednostima, iako oba modela ne mogu predvidjeti šum u nekim dijelovima koji se manifestira kao vrlo visoka/niska vrijednost koja ne pripada trendu.



Dolje je prikazan isječak iz „Evaluation results“ elementa. On nam pruža jednostavan prikaz konačnih rezultata nekih modela. Iz grafa nije bilo moguće vidjeti i predpostaviti koji model ima bolje performanse te iz evaluacije se vidi zašto, postoji relativno malena razlika u točnosti gdje model sa regresijom koristeći neuronsku mrežu ima neznatno bolje performanse od modela linearne regresije.

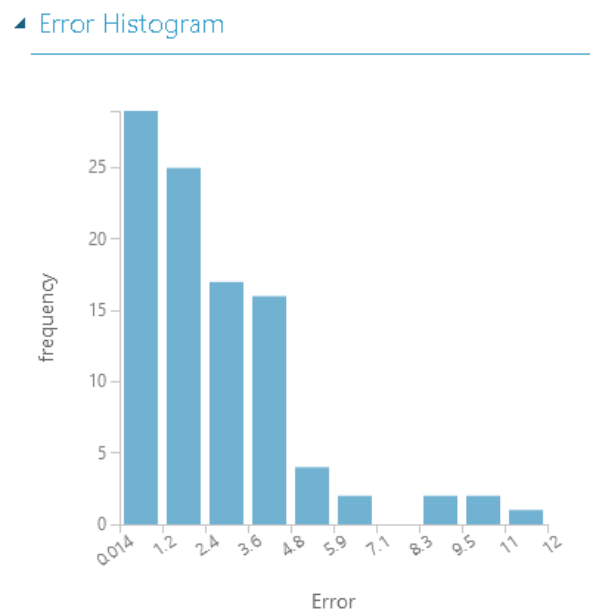
Metrics

Mean Absolute Error	2.871777
Root Mean Squared Error	3.968507
Relative Absolute Error	0.414708
Relative Squared Error	0.232211
Coefficient of Determination	0.767789

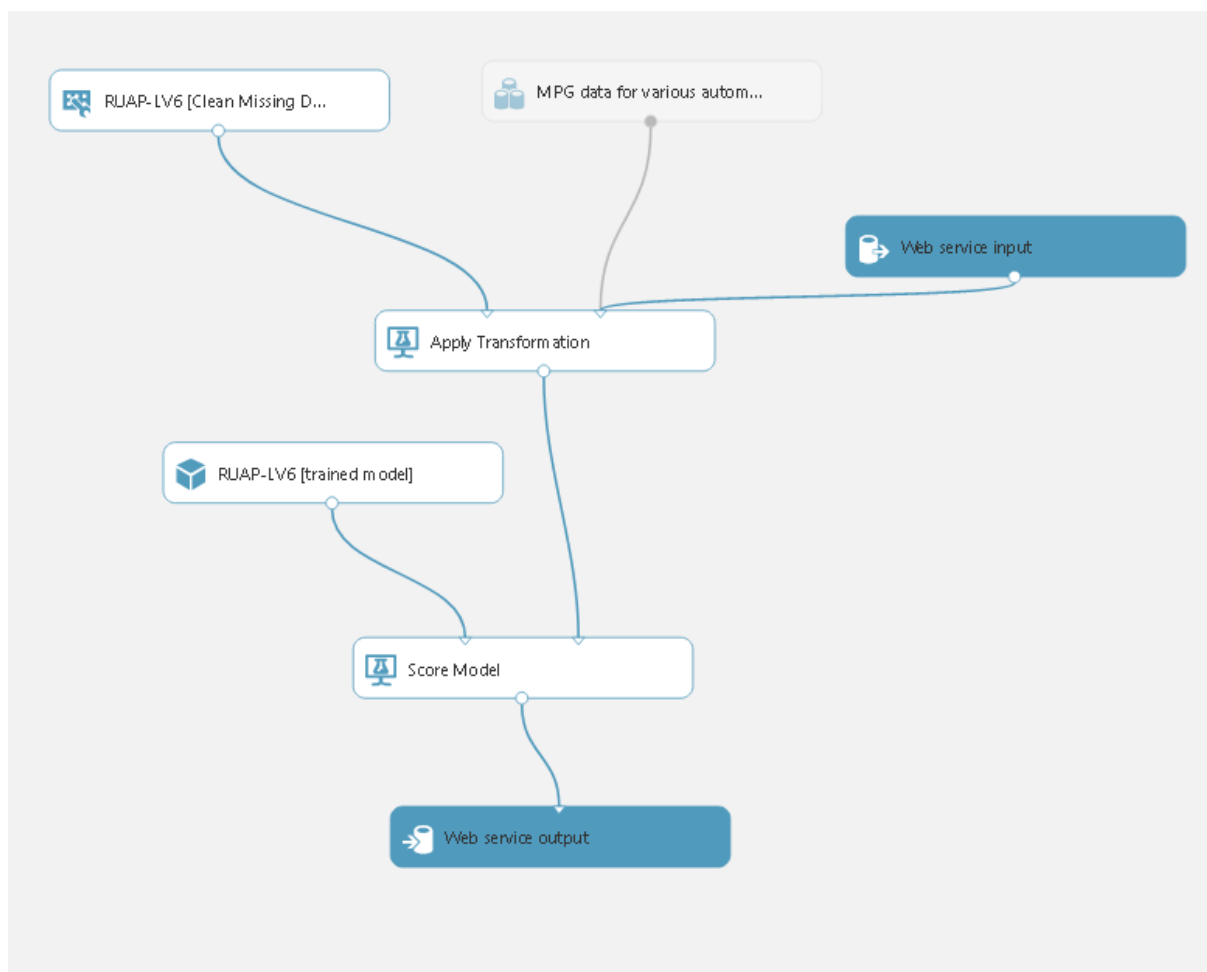


Metrics

Mean Absolute Error	2.646649
Root Mean Squared Error	3.510127
Relative Absolute Error	0.382197
Relative Squared Error	0.181666
Coefficient of Determination	0.818334



Nakon toga smo naš dovršeni model podigli kao web servis.



Te klikom na „Deploy Web Service“ dižemo cijeli sustav online za korištenje.

ruap-lv6 [predictive exp.]

DASHBOARD CONFIGURATION

General [New Web Services Experience](#) **preview**

Published experiment

[View snapshot](#) [View latest](#)

Description

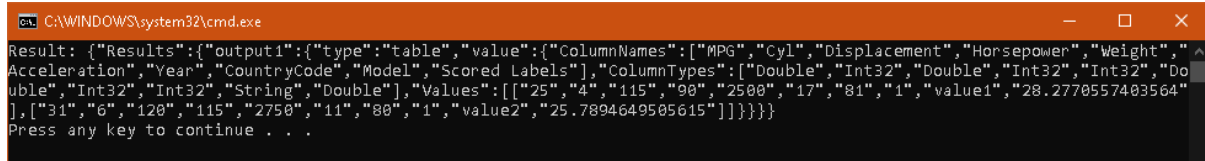
No description provided for this web service.

Sad možemo preko aplikacije pisane u C# poslati i dobivati natrag rezultate.

Npr. koristeći varijable:

```
Values = new string[,] { { "25", "4", "115", "90", "2500", "17", "81", "1", "value1" }, { "31", "6", "120", "115", "2750", "11", "80", "1", "value2" }, }
```

koje šaljemmo dobijemo natrag sljedeći odaziv:

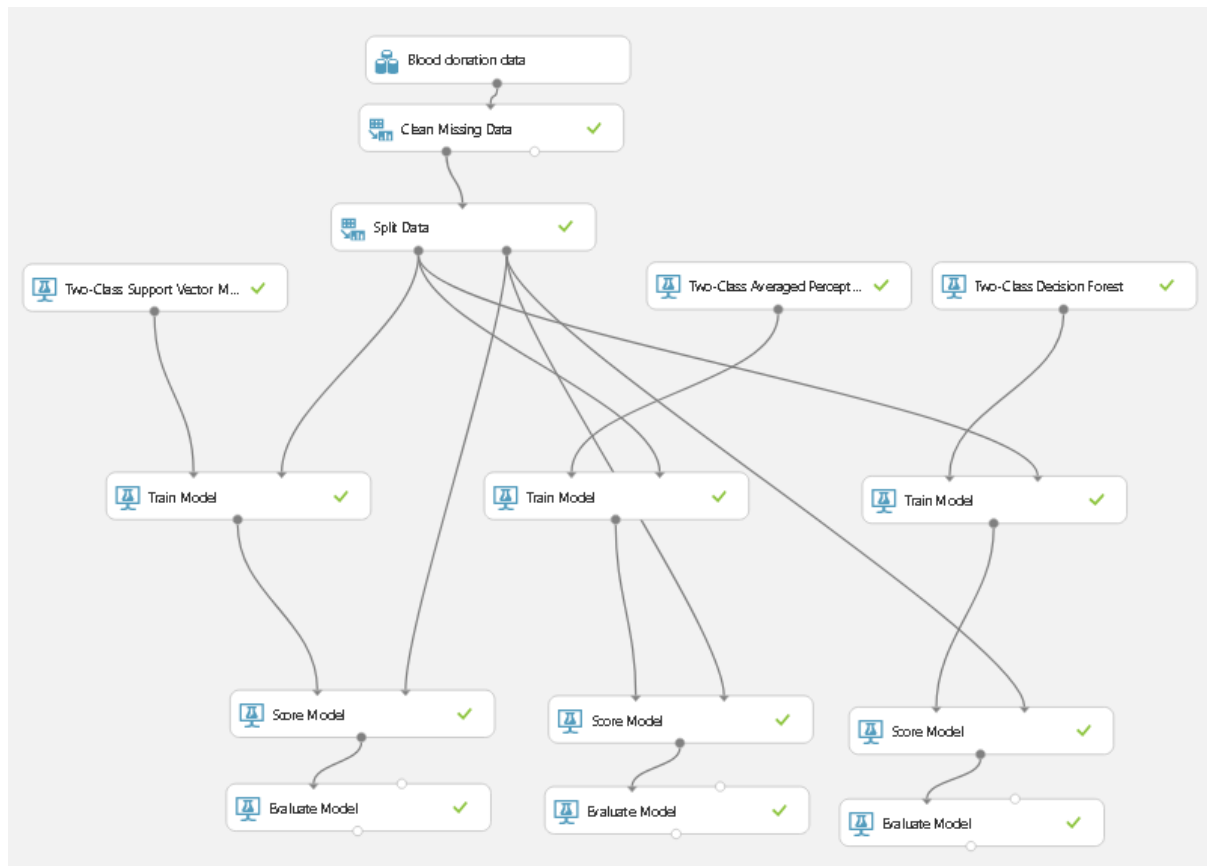


```
C:\WINDOWS\system32\cmd.exe
Result: {"Results":{"output1":{"type":"table","value":{"ColumnNames":["MPG","Cyl","Displacement","Horsepower","Weight","Acceleration","Year","CountryCode","Model","Scored Labels"],"ColumnTypes":["Double","Int32","Double","Int32","Int32","Double","Int32","Int32","String","Double"],"Values":[["25","4","115","90","2500","17","81","1","value1","28.2770557403564"],["31","6","120","115","2750","11","80","1","value2","25.7894649505615"]]]}}}
Press any key to continue . . .
```

gdje nakon vrijednosti imena („value1“ npr.) vidimo rezultat evaluacije.

Zadatak 2.

Moramo slično kao i prethodnom zadatku složiti eksperiment, ali ovaj puta za klasifikaciju skupa podataka o darivanju krvi.



Zadatak nam govori da koristimo normalizaciju da vidimo dali ima utjecaj na normalizaciju podataka zadanog skupa ovisno o različitim parametrima klasifikatora.

Prvo radimo bez normalizacije kao što je prikazano u slici iznad, te koristimo prethodno zadane (default) postavke klasifikatora .

Two-Class Support Vector Machine:

True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.797	0.600	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.150	0.240	

Two-Class Averaged Perceptron:

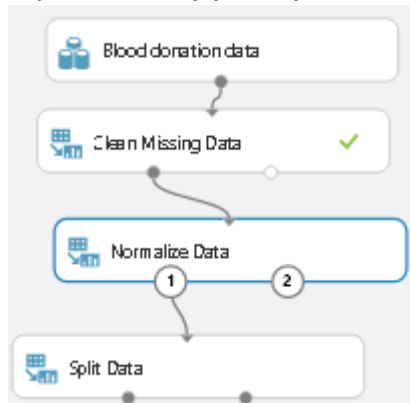
True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.797	0.600	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.150	0.240	

Two-Class Decision Forest:

True Positive	False Negative	Accuracy	Precision	Threshold
15	25	0.786	0.500	0.5
False Positive	True Negative	Recall	F1 Score	
15	132	0.375	0.429	

Prva dva modela su imale iste performanse, dok treći model je imao neznatno goru točnost i preciznost ali imao je veći „Recall“.

Ako upalimo normalizaciju tako da nam sve vrijednosti stupaca svede na jednu normaliziranu granicu vrijednosti dobijaju se sljedeće vrijednosti:



Two-Class Support Vector Machine:

True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.797	0.600	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.150	0.240	

Two-Class Averaged Perceptron:

True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.797	0.600	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.150	0.240	

Two-Class Decision Forest:

True Positive	False Negative	Accuracy	Precision	Threshold
15	25	0.786	0.500	0.5
False Positive	True Negative	Recall	F1 Score	
15	132	0.375	0.429	

Vidimo iz rezultata da normalizacija podataka nema učinka na rezultate koje nam daju klasifikatori.

Sljede rezultati promijene parametara za zadane klasifikatore.

Two-Class Support Vector Machine ima dvije važne varijable: broj iteracija i lambda. Brojem iteracija kontroliramo koliko će dugo klasifikator trenirati ali zauzvrat će imati veću točnost. Lambda služi za fino namještanje klasifikatora. Prethodno smo imali broj iteracija na 1 i lambda na 0.001 .

Broj iteracija:10, lambda:0.001

True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.797	0.600	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.150	0.240	

Broj iteracija:10, lambda:0.01

True Positive	False Negative	Accuracy	Precision	Threshold
5	35	0.791	0.556	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.125	0.204	

Broj iteracija:100, lambda:0.0001

True Positive	False Negative	Accuracy	Precision	Threshold
5	35	0.786	0.500	0.5
False Positive	True Negative	Recall	F1 Score	
5	142	0.125	0.200	

Vidimo da manipuliranjem lambde dobijamo različite rezultate svaki puta, te početno zadani lambda je dao najbolje rezultate. Broj iteracija izgledom nije utjecao na rezultate, moguće zato što i početno zadana vrijednost od 1 je dovoljna za ovakav jednostavan model.

Two-Class Averaged Perceptron ima dvije važne varijable: veličina koraka(engl. Learning rate) i maksimalan broj iteracija. Veličina koraka nam govori koliki korak uzima funkcija u stohastičkom gradijentnom spustu, ako je prenizak algoritam može zapeti u nekom lokalnom minimumu, ako je previsok možemo premašiti pravi globalni minimum. Maksimalni broj iteracija nam daje jednostavnu kontrolu o fittingu: manji broj iteracija nam daje generalno bolji slučaj, dok veći broj iteracija povećava „fit“ funkciji pri opasnosti da prevelik broj iteracija dovodi do overfittinga.

Veličina koraka:1, max broj iteracija:100

True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.791	0.545	0.5
False Positive	True Negative	Recall	F1 Score	
5	142	0.150	0.235	

Veličina koraka:0.5, max broj iteracija:10

True Positive	False Negative	Accuracy	Precision	Threshold
5	35	0.786	0.500	0.5
False Positive	True Negative	Recall	F1 Score	
5	142	0.125	0.200	

Veličina koraka:0.5, max broj iteracija:100

True Positive	False Negative	Accuracy	Precision	Threshold
5	35	0.786	0.500	0.5
False Positive	True Negative	Recall	F1 Score	
5	142	0.125	0.200	

Veličina koraka:5, max broj iteracija:10

True Positive	False Negative	Accuracy	Precision	Threshold
6	34	0.797	0.600	0.5
False Positive	True Negative	Recall	F1 Score	
4	143	0.150	0.240	

Vidimo da promjenom parametara mijenjamo ponašanje algoritma pri treniranju. Kao i očekivano, prevelik broj iteracija dovodi do overfittinga koje dovodi do mogućih grešaka pri procjeni podataka. Veličina koraka isto tako utječe na ponašanje, te prepoznamo da namještanjem pogrešnih vrijednosti varijabli dovodi do promijene preciznosti, točnosti i „Recall“-a modela.

Two-Class Decision Forest ima dvije važne varijable: broj stabala odluka (engl. Decision Trees) i maksimalna dubina stabala. Sa većim brojem drveta povećavamo raspon mogućih varijabli koje će primiti ali povećavamo vrijeme treniranja. Maksimalnom dubinom drveta kontroliramo preciznost, ali i povećavamo rizik overfittinga.

Broj drveta odluke:16, max dubina:32

True Positive	False Negative	Accuracy	Precision	Threshold
14	26	0.791	0.519	0.5
False Positive	True Negative	Recall	F1 Score	
13	134	0.350	0.418	

Broj drveta odluke: 64, max dubina:32

True Positive	False Negative	Accuracy	Precision	Threshold
15	25	0.797	0.536	0.5
False Positive	True Negative	Recall	F1 Score	
13	134	0.375	0.441	

Broj drveta odluke:8, max dubina: 128

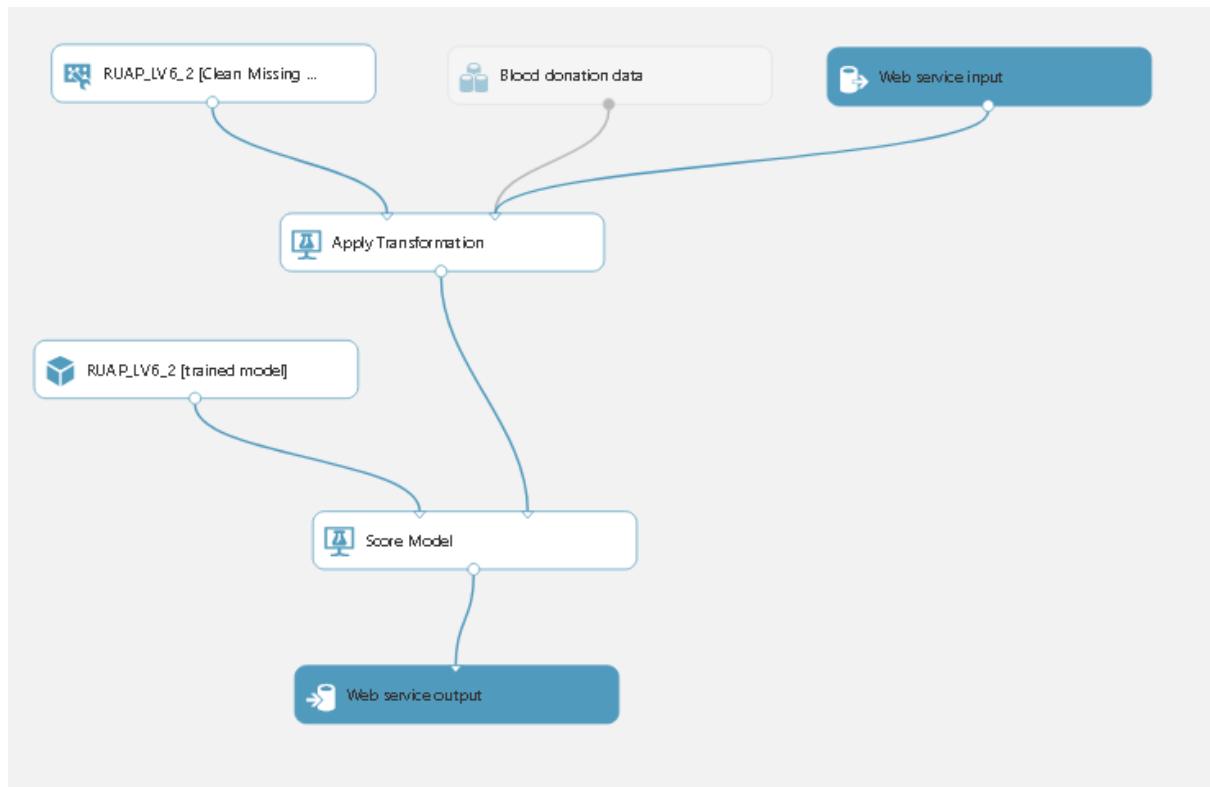
True Positive	False Negative	Accuracy	Precision	Threshold
15	25	0.786	0.500	0.5
False Positive	True Negative	Recall	F1 Score	
15	132	0.375	0.429	

Broj drveta odluke:32, max dubina:64

True Positive	False Negative	Accuracy	Precision	Threshold
15	25	0.797	0.536	0.5
False Positive	True Negative	Recall	F1 Score	
13	134	0.375	0.441	

Vidimo da promijenom broja drveta nam dobro služi jer povećava kvalitetu rezultata, ali i osjetno povećava dužinu trajanja treninga. Povećanjem dubine ne dovodi do znatnih promijena, moguće jer je premalen broj uzoraka u testnom skupu.

Utvrdili smo da je najpogodniji klasifikator Two-Class Decision Tree jer ima neznatno goru preciznost i točnost ali ima znatno bolji „Recall“. Klikom na odabrani „Train model“ te onda na „Set up web service“ namještamo model za dizanje na web.



Klikom na „Run“ pa na „Deploy web service“ dižemo sustav na web.

ruap_lv6_2 [predictive exp.]

DASHBOARD CONFIGURATION

General [New Web Services Experience](#) **preview**

Published experiment

[View snapshot](#) [View latest](#)

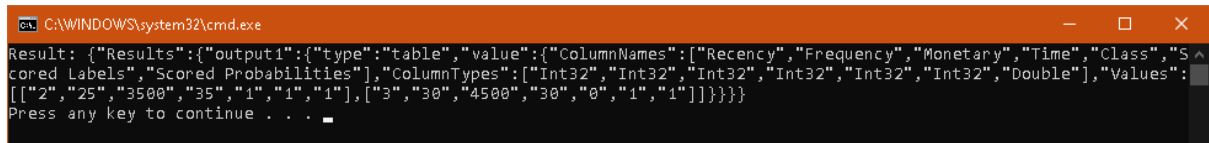
Description

No description provided for this web service.

Koristeći varijable:

```
ColumnNames = new string[] { "Recency", "Frequency", "Monetary", "Time", "Class"},  
Values = new string[,] { { "2", "25", "3500", "35", "1" }, { "3", "30", "4500",  
"30", "0" }, }
```

Dobijamo sljedeći odaziv od web usluge:



```
CS: C:\WINDOWS\system32\cmd.exe  
Result: {"Results":{"output1":{"type":"table","value":{"ColumnNames":["Recency","Frequency","Monetary","Time","Class","Scored Labels","Scored Probabilities"],"ColumnTypes":["Int32","Int32","Int32","Int32","Int32","Int32","Double"],"Values":  
[[["2","25","3500","35","1","1"],["3","30","4500","30","0","1"]]]}}}  
Press any key to continue . . .
```

Gdje nam zadnja varijabla u dobivenim poljima predstavlja predviđenu vrijednost klase.