Kunskapskontroll 2

Machine learning ds23



William Blennow

EC Utbildning

Examinationsuppgift

2024-03

# Abstract

The following report is a machine learning documentation based on the MNIST data set. In the report different classification models are trained on the existing data set and compared in their accuracy score to make a selection of the best performing model. Then the selected model is used to make predictions on pictures of hand written numbers taken by a mobile phone. The uploaded pictures are processed to identify numbers and then the identified numbers are processed to match the MNIST data set and the selected model is used to predict what numbers are on the picture.

Innehållsförteckning

[Abstract 2](#_Toc161846547)

[1 Inledning 1](#_Toc161846548)

[1.1 Underrubrik – Exempel 1](#_Toc161846549)

[2 Teori 2](#_Toc161846550)

[2.1 MNIST dataset 2](#_Toc161846551)

[2.2 Exempel: Klassifikationsmodeller 2](#_Toc161846552)

[2.2.1 Exempel: K Neighbours Classifier 2](#_Toc161846553)

[2.2.2 Exempel: Support Vector Machines 2](#_Toc161846554)

[2.2.3 Exempel: Random Forest Classifier 2](#_Toc161846555)

[2.2 Exempel: Neurala Nätverk 2](#_Toc161846556)

[3 Metod 3](#_Toc161846557)

[4 Resultat och Diskussion 4](#_Toc161846558)

[5 Slutsatser 5](#_Toc161846559)

[6 Teoretiska frågor 6](#_Toc161846560)

[7 Självutvärdering 8](#_Toc161846561)

[Appendix A 9](#_Toc161846562)

[Källförteckning 10](#_Toc161846563)

# Inledning

Syftet med denna rapport är att använda maskininlärning för att modellera MNIST-data och utvärdera denna genom olika modeller.

Detta ska göras genom ett komplett ML-flöde från början till slut, i detta så ingår att ladda in data, optimera hyperparametrar, validera de valda modellerna, utvärdera modellerna för att sedan välja en modell och utvärdera den gentemot test-datan.

Därefter så ska denna modell användas i en streamlitapplikation där bilder transformeras, och behandlas så att den valda modellen kan användas för att göra prediktioner på siffror som identifieras på bilder som är tagna med en mobiltelefon.

För att uppfylla syftet så kommer följande frågeställning(ar) att besvaras:

1. Vilken av de valda klassifikationsmodellerna presterar bäst på MNIST-data?
2. Hur behöver bilder behandlas för att kunna använda den bäst presterande modellen för att prediktera egna handskrivna siffror?

Syftet är inte att skapa världens bäst presterande modell på MNIST eller en webapplikation som är helt perfekt. Detta hade introducerat betydligt mer komplexa modeller och framförallt ytterligare modeller inom maskininlärning och deep learning för att optimera bildbehandlingen. Utan det huvudsakliga i denna rapport är att sy ihop modellval och sedan att kunna använda en intränad modell i ett reellt syfte, ”ta ut experimentet från labbet till verkligheten”.

# Teori

## MNIST dataset

MNIST är ett dataset som består av 70.000 bilder på handskrivna siffror, skrivna av gymnasiestudenter och offentligt anställda i USA. Varje bild har en etikett kopplad till bilden som beskriver vilken siffra som är på bilden. (Géron, 2019, s 85)

Bilderna transformeras till en matris på 28\*28 pixlar där varje pixel anges ett värde som indikerar färgen på varje pixel i matrisen. Om vi vill träna modeller för att predicera vilken etikett bilden har baserat på matriserna så har vi ett klassifikationsproblem med 10 olika klasser.

## Exempel: Klassifikationsmodeller

Då det är ett klassifikationsproblem som vi vill träna våra modeller på så utgår jag från modeller som är anpassade för detta, det finns en rad olika av dessa men jag har valt att använda tre styckenoch att optimera vissa valda hyperparametrar för dem. Det som modellerna har gemensamt är att de används för att lösa klassifikationsproblem, dvs problem där vår oberoende variabel har ett diskret värde och inte ett kontinuerligt sådant. I det fallet hade jag använt mig av regressionsmodeller.

### Exempel: K Neighbours Classifier

Denna modell är baserad på vad observationer som gränsar till varandra har för värde, den använder avståndet mellan observationerna för att skapa en modell där den predicerar utifrån avstånd på de nya observationerna i jämförelse med vad den har tränats på.

Modellen har, likt många andra. En rad olika hyperparametrar: **Neighbours** är antalet grannar som man väljer att använda för att göra prediktionen. **Weight** där man anger huruvida man viktar närmre observationer mer eller om alla grannar viktas lika mycket. **Algorithm** anger vilken typ av algoritm som används för att räkna ut avståndet och **P** som styr hur man mäter avstånden. (Scikitlearn, 2024)

### Exempel: Support Vector Machines

I klassifikationsproblem så fungerar Support Vector Machines fungerar genom att skapa så stora avstånd mellan de olika klasserna för att sedan kunna göra prediktioner utifrån observationer och hur deras avståndsförhållanden ser ut.

SVM kan användas för både regression och klassifikationsproblem, (Géron, 2019, s 153) vid klassifikationsproblem så finns det en rad olika modeller som baseras på SVM, men en av de mer använda är linjär SVC (Support Vector Classifier)

SVC är inte nödvändigtvis lämpat för större dataset (Scikitlearn, 2024) då det kräver mycket processorkraft att optimera modeller som är baserade på SVM men kan vara oerhört kraftfulla om hyperparametrarna optimeras på rätt sätt. Några av hyperparametrarna för SVC är **C,** vilket regulariserar datan, desto lägre siffran är desto mer regulariserad blir modellen. Vi har även **Kernel** som kan göra modellen ickelinjär om det passar datan bättre (Scikitlearn, 2024). Linjär SVC är mindre resurskrävande än övriga kernels och mer lämpad för större dataset. Vi har också **Gamma** som påverkar koefficienten när man använder kernels som inte är ”linear”.

### Exempel: Random Forest Classifier

Random Forest Classifier är en ensemblemetod baserad på beslutsträd med slumpmässiga urval, det man gör är att träna flera olika grupper på delar av träningsdatan för att sedan genom dessa skapa en väldigt rigid och kraftfull modell. (Géron, 2019, s 189) Genom att skapa flera olika modeller med beslutsträd och sedan jämföra de modellerna mot varandra så identifieras den bästa modellen av dem.

Det finns som med de flesta modeller även flera hyperparametrar. Den främsta är **n\_estimators** vilken anger hur många beslutsträd som ingår i modellen, **max\_features** är antalet features som tas i beaktning för att göra en split i beslutsträdet. **Max\_depth** anger hur djupt beslutsträdet kan bli, **min\_samples\_leaf** och **min\_samples\_split** anger hur många observationer som minst krävs för att nå till slutet av en gren och split hur många som minst krävs för att gå vidare i beslutsträdet. **Bootstrap** anger om datan i träden sker med eller utan återläggning. (ScikitLearn, 2024)

### Exempel: Cross Validation och Accuracy Score

För att optimera modeller och deras hyperparametrar så kan man med fördel göra en cross validation, det finns flera olika sätt att göra detta men en användbar metod är att använda sig av observationerna och testa olika hyperparametrar, detta går att göra genom GridsearchCV, då testar man de olika hyperparametrarna genom att korsvalidera dem i de olika angivna kombinationerna, vilket kan vara rätt resurskrävande. (ScikitLearn, 2024) För att göra detta mer effektivt kan man använda sig av RandomizedsearchCV för att skapa slumpmässiga kombinationer och se vilka av de olika hyperparametrarna som presterar bäst.

Accuracy score är en metod för att se hur modellen presterar, i ett klassifikationsproblem så görs detta genom att mäta antalet korrekta prediktioner mot det faktiska resultatet och återger ett värde mellan 1 (perfekt resultat) och 0 (inga korrekta prediktioner). (ScikitLearn, 2024)

# Metod

Genomförandet av arbetet är uppdelat i två delar, den första delen av metoden svarar på frågeställning 1 och den andra delen för att svara på frågeställning 2

## Metod för modellering av prediktor

För att skapa min modell så har jag först laddat in datasetet MNIST där X = matriserna med bilddatan och Y = den oberoende variabeln som vi vill kunna predicera.

Jag har valt de tre olika modellerna K Neighbors Classifier, SVC och Random Forest Classifier

Först gör jag ett mindre urval datan för att göra en preliminär optimering av hyperparametrar för att spara på processortiden som krävs. Denna cross validation gör jag genom Randomized Search CV som istället för att gå igenom alla kombinationer gör slumpmässiga urval och kombinationer för att validera de olika hyperparametrarna.

Vid detta urval så normaliserar jag också datan för modellerna KNN och SVC för att de ska prestera optimalt.

När jag nu har en idé om vilka hyperparametrar som passar för att lösa problemet så delar jag upp hela testsetet i test, validering och tränings – data, jag skapar också normaliserade värden på dessa utifrån att våra två modeller som gynnas av det ska ha så goda förutsättningar som möjligt.

Sedan skapar jag param grids där våra olika hyperparametrar som identifierats blir avgränsade för att sedan genomföra en komplett grid search CV på hela träningssetet

När modellerna har fått sina optimala hyperparametrar genom denna cross validation så jämför jag hur de presterar på accuracy score gentemot valideringsdatan. Eftersom accuracy score och F1 score är samma i detta klassifikationsproblem anser jag det tillräckligt.

Sedan har den bäst presterande modellen valts (Random Forest Classifier) och då tränas denna på träning och valideringsdatan för att sedan mätas mot test-datan. Då accuracy score förblir på samma höga nivå så är jag nöjd med detta resultatet och denna modell och sparar den för att kunna använda den som prediktor i min webapplikation

## Metod för bildbehandling och prediktion

Den sparade prediktorn laddas in i applikationen.

Därefter defineras en klass som heter Prediction som innehar bilden som ska prediceras och funktioner för att behandla bilden så att den är i samma format som datan från MNIST. Även funktioner för prediktion och hur säker min modell är för prediktionen ingår i denna klass.

Efter detta så skapas det en funktion för att behandla bilden, detta sker genom openCV som gör om bilden till gråskala om det behövs, och genom tröskelvärden så söker den efter konturer i bilden.

För varje kontur som är större än den angivna minsta arean för konturerna så skapar vi en bild genom att skapa en kvadrat med visst utrymme mot kanterna så att den identifierade konturen behåller sin dimension när den sedan ska behandlas och prediceras, därefter så läggs denna bild till i en lista som efter att bilden bearbetas innehåller nya formaterade bilder på alla identifierade konturer som är större än den angivna miniarean.

Sedan så skapas en enkel streamlitapplikation där man kan ladda upp en bild, denna bild öppnas med PIL som också gör om storleken på den men bibehåller dimensionerna. Bilden behandlas till en array genom numpy och därefter så kallar vi på funktionen för att identifiera konturer och spara dem i listan.

Ursprungsbilden visas och sedan så följer en for-loop för varje identifierad kontur där det kallas på prediktionen (detta innebär att vår sparade kvadratiska bild omvandlas till att matcha MNIST-data) och sannolikheten för denna prediktion. Sedan så visas både den sparade kvadraten och den processade (MNIST-kompatibla) bilden upp och vår prediktion samt säkerheten vår modell har för den prediktionen skrivs ut.

# Resultat och Diskussion

## Modellval

Efter att de valda modellernas hyperparametrar har optimerats genom Grid Search CV så testas deras accuracy score på valideringsdatan vilket ger högst resultat för random forest classifier-modellen

|  |  |
| --- | --- |
| **Accuracy score för olika modeller** | |
| K Neighbors classifer | 0.9597 |
| SVC | 0.9292 |
| Random Forest Classifier | 0.9745 |

Tabell 1: Accuracy Score för de tre valda modellerna på valideringsdata.

Efter att ha valt Random Forest Classifier som vår bäst presterande modell så tränas den på tränings och valideringsdatan och utvärderas på testdatan vilket ger en accuracy score på 0.971, det vill säga vår tränade och valda modell presterar i linje med förväntningarna till en träffsäkerhet på 97,1%. Denna prestation bedömer jag är tillräckligt bra för att gå vidare med och skapa en applikation som kan predicera handskrivna siffror.

## Bildbehandling och prediktion

Genom att skapa funktioner som identifierar konturer och sedan behandla dessa identifierade konturer i linje med MNIST-datan så ser vi att vår applikation lyckas identifiera 3 siffror och predicera dem med en sannolikhet på 84,4%, 82,60% och 69.39%. Vi kan också se att bilderna har behandlats utifrån originalbilden och konverterats i linje med MNIST-datan, se appendix A.

# Slutsatser

## Vilken av de valda modellerna presterar bäst på MNIST-data

Av de valda modellerna så presterar Random Forest Classifier bäst, men det är också den modellen i modellvalet som får flest hyperparametrar optimerade. Det finns en tydlig bias i modellvalet där de andra modellerna inte getts lika mycket processorkraft för att optimeras. Anledningen till detta är mina tidigare observationer av datasetet och min erfarenhet av modellens prestation utan några justerade hyperparametrar och dess träffsäkerhet.

Mitt val av KNN och SVC var medvetna då de skiljer sig från varandra och den tredje modellen. Jag är övertygad om att SVC med en annan kernel och optimering av ytterligare hyperparametrar hade kunnat prestera bättre men det hade krävt otroligt mycket mer processorkraft än vad som hade varit lämpligt för att hitta en tillfredsställande modell. Valet av SVC med färre hyperparametrar baserades på att SVM inte är jättelämpliga på större dataset och genomförandet av denna process har gett mig mer insikter kring detta.

Generellt sett så kan man säga att skillnaden på 0.9597 och 0.9745 är marginell och har gett oss modeller som presterar tillräckligt precist för att kunna genomföra steg 2 i rapporten, men för att säkerställa en korrekt metod så har jag gjort en jämförelse och genomfört samma steg i modellvalet.

## Hur behöver bilder behandlas för att kunna använda den bäst presterande modellen för att prediktera egna handskrivna siffror?

För att lyckas göra detta så krävs det först att identifiera områden på bilden som har siffror. I min studie så har jag valt att göra en ganska simpel modell, hade denna utsatts för mer brus, ex bilder med flera lager av konturer eller andra ickestandardiserade bilder. Exempelvis om andra föremål syns i bakgrunden, eller siffror skrivna i negativ yta så kommer inte funktionerna kunna predicera någonting av värde.

Applikationen är en grund och introduktion till bildbehandling för att applicera maskininlärning och den har uppnått syftet som har konkretiserats från rapportens inledning.

För att vidare besvara syftet så förutom identifikationen krävs det sedan att vidare processa den identifierade konturen för att matcha datan som den valda modellen är tränad på som den är tränad på. Detta görs genom att konvertera storleken på bilden, och inventera den så att den är likvärdig med testdatan.

Just konverteringen är inte det som ger högst precision och användning av applikationen utan är mer ett generellt steg. För att applikationen ska bli mer användbar och robust så krävs det fler steg i funktionen och bildbehandlingen. Genom ytterligare maskininlärning och eventuell deep learning hade detta kunnat göras mer precist men hade också varit överflödigt då detta inte varit syftet med rapporten.

# Teoretiska frågor

1. **Kalle delar upp sin data i ”Träning”, ”Validering” och ”Test”, vad används respektive del för?**

**Träningsdatan** används för att träna upp olika modeller på datasetet. **Valideringsdatan** används sedan för att jämföra de olika modellerna på en del av datan som modellerna inte har tränats på för att besluta vilken modell som man sedan tränar på träning + valideringsdatan för att sedan utvärdera på **testdatan**.

1. Julia delar upp sin data i träning och test. På träningsdatan så tränar hon tre modeller; ”Linjär Regression”, ”Lasso regression” och en ”Random Forest modell”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”valideringsdataset”?

Julia behöver göra en cross validation på sina modeller, detta kan hon göra genom att använda K-fold cross validation där man slumpmässigt delar upp träningsdatan i K delar. Sedan så tränar använder man alla delar förutom en för att träna modellen och använder den återstående för att validera modellen. Detta upprepas K gånger där varje del agerar valideringsdata. Då får man ett MSE på varje valideringsdel och sedan så tar man MSE/K för att få ett genomsnittsfel genom sin validering.

Genom att göra detta så kan Julia välja den modell som presterar bäst i cross validation för att testa på sin testdata.

1. **Vad är ”regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?**

Regressionsproblem är när man vill studera variabler som är kontinuerliga dvs. variabler som kan anta oändligt många olika värden, exempel på detta är längd och vikt på nyfödda barn (då vi med precisa verktyg kan mäta dessa med precision som gör att varje observation rent teoretiskt skulle kunna anta ett unikt värde).

Modeller som kan användas inom regressionsproblem är t.ex linjär regression, ridge regression, lasso regression och random forest.

Potentiella tillämpningsområden skulle då kunna vara att studera nyfödda barns vikt i förhållande till längd och jämföra mot parametrar kring deras föräldrars inkomst, ålder, längd och vikt.

1. **Hur kan du tolka RMSE och vad används det till: 𝑅𝑀𝑆𝐸 = √ 1 𝑛 ∑(𝑦𝑖 − 𝑦̂𝑖 ) 2 𝑛 1**

RMSE (root mean squared error) är roten av MSE mean squared error, dvs vårt genomsnittsfel. Anledningen till att man använder RMSE är för att MSE så kvadrerar vi våra fel för att vi ska kunna jämföra positiva tal, och därmed hitta genomsnittsfelet (om vi enbart hade summerat värdena så kan vi ha både positiva och negativa värden och inte få en korrekt bild). Därefter behöver vi ta roten ur MSE för att ha ett värde som är jämförbart med vår ursprungsvariabel.

1. **Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”?**

Klassificeringsproblem är, till skillnad från regressionsproblem, problem där vi har diskreta värden. Man skulle kunna beskriva dem som olika kategorier eller binära tal.

Modeller som kan användas för klassificeringsproblem är b.la. logistisk regression, K neighbors classifier eller random forest classifier. Dessa kan vi använda om vi vill t.ex ta reda på om en kund churnar, eller om en röd jacka förväntas sälja bättre än en gul eller grön jacka.

En confusion matrix är en tabell där vi kan se vad hur vår modell presterar genom att se vilka klasser som den predikterar rätt på (true positive, true negative) och också vilka den predikterar fel på (false positive, false negative)

1. **Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.**

K-means är en modell som används för att göra prediktioner genom att klustra datan, dvs att detta är en sk ”unuspervised”-modell. Den gör detta genom att identifiera centroids = antal K och sen allokerar den alla observationer till de centroids som ”i mitten” av de olika klustren. Means innebär att klustrerna skapas genom att mäta avstånd till centroids så indelningen i kluster beror på avstånden till de identifierade centroidsen av våra kluster.

Detta kan exempelvis användas för att segmentera kunder i olika grupper om man vill anpassa information eller marknadsföringskampanjer till dessa.

1. **Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding. Se mappen ”l8” på GitHub om du behöver repetition.**

Kategorisk data, tex färger eller nationalitet, biltillverkare etc, har ingen inbördes ordning utan denna måste ges värden. För att göra det så kan vi använda oss av dessa 3 metoder:

Ordinal encoding så ger vi vår kategoriska data en siffra tex. dansk = 0, norsk = 1, serb = 2. Detta skapar också en rangordning mellan vår data, i detta fallet baserat på alfabetisk ordning

One-hot encoding så skapar vi en binär array där vi i vårt fall har 3 olika binära koder där dansk = [1, 0, 0], norsk = [0, 1, 0] och serb = [0, 0, 1].

Dummy variable encoding är likt one-hot enconding men iom att vi vet att vi har 3 kategorier så kan vi utesluta den sista (n-1 där n = antal kategorier). Detta för att inte skapa matriser som är singulära och inte kunna utföra linjär algebra på, i detta exempel: dansk = [1, 0], norsk = [0, 1], serb = [0, 0].

1. **Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {röd, grön, blå} generellt sett inte har någon inbördes ordning (nominal) men om du har en röd skjorta så är du vackrast på festen (ordinal) – vem har rätt?**

Julia har rätt, men det hon har gjort i sitt exempel är att ange en inbördes rangordning mellan de nominala värdena ”röd, grön, blå” till en ordinal rangordning. All data behöver ju tolkas och för att exempelvis använda modeller på dessa så behöver vi bearbeta datan som i svaret på föregående fråga (fråga 7), detta betyder inte att röd alltid i bäst men i Julias modell för att ”klä sig på fest” så är röd skjorta rangordnat högst i relation till de andra färgerna på skjortor.

1. **Kolla följande video om Streamlit: https://www.youtube.com/watch?v=ggDaRzPP7A&list=PLgzaMbMPEHEx9Als3F3sKKXexWnyEKH45&index=12 Och besvara följande fråga: - Vad är Streamlit för något och vad kan det användas till?**

Streamlit är ett open source python bibliotek som gör att man kan skapa webapplikationer och publicera dem genom streamlit så att man kan dela sina script med andra.

# Självutvärdering

1. Utmaningar du haft under arbetet samt hur du hanterat dem.

Jag tycker den största utmaningen har varit att hitta nivån på hur uttömmande svaren ska vara och hur mycket teori som ska presenteras. Efter att ha läst igenom uppgiften, studiehandledningen och övrig dokumentation så har jag landat i att det primära är genomförandet av uppgiften och att presentera teori och diskutera metod är mer sekundärt. Jag har fått påminna mig om att det är en rapport och inte en uppsats vilket har varit lite otydligt, iaf för mig. Men jag tror och hoppas att nivån är på en korrekt nivå.

1. Vilket betyg du anser att du skall ha och varför.

Jag upplever att jag har lyckats med G och VG-delen.

Jag upplever också att jag i mina slutsatser resonerar kring min process och tillkortakommanden i denna samt generell applicering av applikationen som har framtagits. Vilket uppfyller kravet för redogörning och kritisk diskussion kring modellval, modellanpassning och modellutvärdering.

Jag tycker att jag har angripit problemet på ett korrekt och metodiskt sätt med hög säkerhet.

Så min samlade bedömning är att jag har nått upp till samtliga krav som definierats på förhand för VG

1. Något du vill lyfta fram till Antonio?

Jag tycker det har varit en kul och inspirerande kurs, däremot så har det varit lite skillnad på nivå mellan studenterna och delaktighetsgrad på lektionerna, vissa av de grupperna har gett mig väldigt mycket och (tyvärr vid fler tillfällen än de andra) så har det mest känts som slöseri med tid där det uppvisas väldigt lite intresse och engagemang i grupperna. Väldigt kul att applicera teorin på ett verkligt problem. Lite svårt med scopet och omfattningen på rapporten och huruvida formalia har varit viktigare än genomförandet av uppgiften.

# Appendix A

En bild som visar text, handskrift, Teckensnitt, kalligrafi

Automatiskt genererad beskrivning

En bild som visar text, Teckensnitt, grafisk design, vit

Automatiskt genererad beskrivning

# Källförteckning

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. <http://14.139.161.31/OddSem-0822-1122/Hands-On_Machine_Learning_with_Scikit-Learn-Keras-and-TensorFlow-2nd-Edition-Aurelien-Geron.pdf>

ScikitLearn. (22 mars 2024). Nearest Neighbors. <https://scikit-learn.org/stable/modules/neighbors.html#classification.index.html>

ScikitLearn. (22 mars 2024). SVC. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

ScikitLearn (22 mars 2024). Plot classification boundaries with different SVM Kernels. <https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py>

ScikitLearn (22 mars 2024). RandomForestClassifier. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

ScikitLearn (22 mars 2024). Tuning the hyper-parameters of an estimator. <https://scikit-learn.org/stable/modules/grid_search.html>

ScikitLearn (22 mars 2024). 3.3.2.2. Accuracy score.

<https://scikit-learn.org/stable/modules/model_evaluation.html#metrics-and-scoring-quantifying-the-quality-of-predictions.html>