Maskininlärningsprojekt

MNIST handskriven siffror igenkänning

Student: Muhammad Mahmudur Rahman

Kurs: Maskininlärning

Kurslärare: Antonio Prgomet

Utbildning: Data Scientist

Skola: EC Utbildning, Stockholm, Sweden

Datum: 2024-03-22

Abstract

The scope of machine learning (ML) is increasing day by day nowadays. To work in the field of machine learning, handwriting recognition is one of the examples in daily life. In this project, MNIST dataset which are handwritten digits have been used to evaluate the performance comparison of three different machine learning models namely Support Vector Machine (SVM), Random Forest and Logistic Regression where Logistic Regression model showed more accurate predicted results out of them. This project was implemented through popular Python programing language and powerful scikit-learn library.

Innehållsförteckning

1 Inledning

Inom det dynamiska och ständigt föränderliga området för artificiell intelligens framstår maskininlärning som en revolutionerande kraft som driver framsteg inom många applikationer – från röstigenkänningssystem till självkörande bilar. Ett av tillämpningsområdena är handskriftsigenkänning som ett växande område. Handskriftsigenkänning möjliggör kommunikation mellan maskiner och människor och är ett område som syftar till att underlätta denna kommunikation. Detta projekt dyker ner i den praktiska implementeringen av maskininlärning och visar deras kraft och mångsidighet i bildigenkänningsuppgifter med hjälp av den ikoniska MNIST-datauppsättningen.

MNIST-datauppsättningen, en samling handskrivna siffror, har varit riktmärket för bildklassificeringsalgoritmer, vilket ger en lekplats för både nybörjare och experter att testa gränserna för algoritmisk noggrannhet. MNIST-datauppsättningen innehåller 70 000 bilder (vardera 28x28 pixlar) konverterade till en dimension. Som standard är hela datasetet förindelat i 60 000 träningsuppsättningar och 10 000 testset.[1]

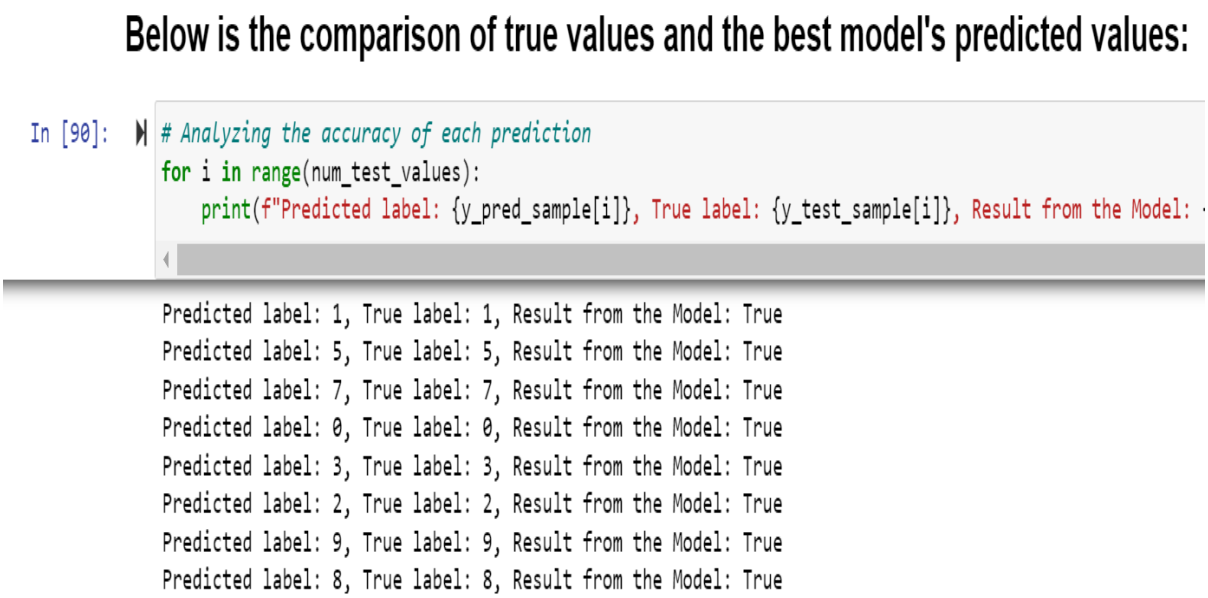
Figur 1. MNIST Dataset exempel[1]

(Källa: Articles for Machine Learning and AI using Python av Rany ElHousieny, PhD på LinkedIn.com)

1.1 Syfte

Detta projekt syftar till att känna igen de handskrivna siffrorna genom att använda tre olika maskininlärningsalgoritmer för att träna klassificerarna med fokus på användning av Python och scikit-learn-biblioteket, så det ger en hög igenkänningsprestanda som vi kan se på bilden nedan. Inom maskininlärning är klassificering problemet med att identifiera vilken kategori en ny observation tillhör. I det här projektet kommer vi att sträva efter att korrekt identifiera siffrorna från dessa binära bilder. Vi kommer att använda logistisk regression för att klassificera varje bild till dess målvärde för sanna siffror. Dessutom kommer vi att jämföra dess prestanda med en annan klassificeringsalgoritm som är Support Vector Machine (SVM) och Random Forest.[2]

(Källa: MNIST Dataset Classification av Atif Ahmed på LinkedIn.com)[2]

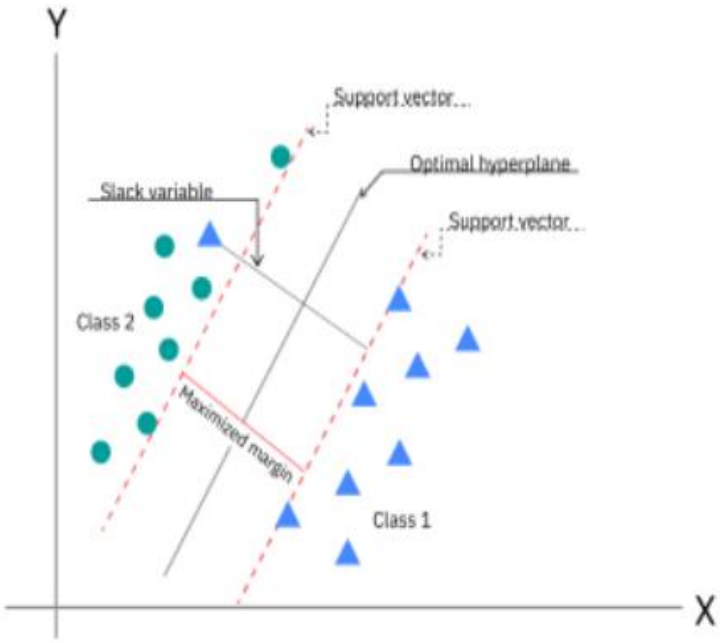


Figur 2. jämförelse av sanna värden och den bästa modellens förutsagda värden

2 Teori

2.1 Support Vector Machine (SVM)

En stödvektormaskin (SVM) är en övervakad maskininlärningsalgoritm som klassificerar data genom att hitta en optimal linje eller hyperplan som maximerar avståndet mellan varje klass i ett N-dimensionellt utrymme. SVM-algoritmen används ofta inom maskininlärning eftersom den kan hantera både linjära och olinjära klassificeringsuppgifter.[3]



Figur 3. Support Vector Machine (SVM)[3]

(Källa: Support Vector Machine på IBM.com)[3]

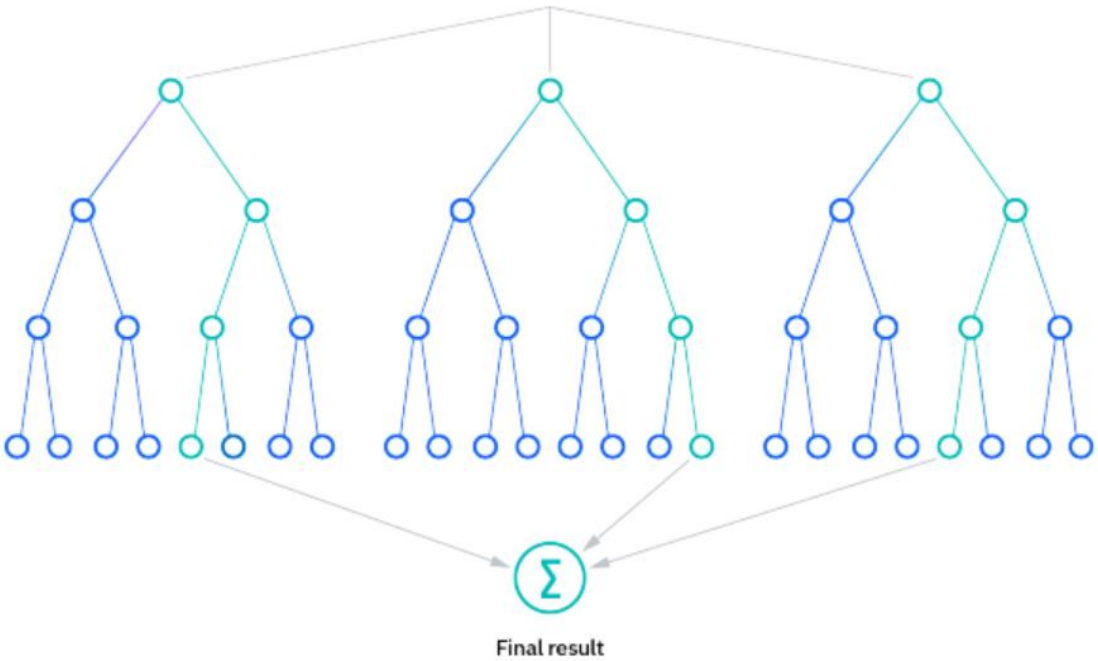
2.1.1 Hyperparameter för Support Vector Machine (SVM)

SVM hyperparametrar är de inställningar som vi kan justera innan vi tränar modellen och påverka hur modellen passar data. Vanligtvis inkluderar dessa kärnfunktionen, som mappar data till ett högre dimensionellt utrymme där en linjär beslutsgräns kan hittas. Det finns olika typer av kärnor, som linjära, polynomiska, radiella basfunktioner (RBF) och sigmoid. Dessutom finns det regulariseringsparametern, eller C, som balanserar mellan att maximera marginalen och minimera träningsfelet.[4]

(Källa: "How do you optimize the hyperparameters of SVM for industrial classification problems?" av AI and the LinkedIn community på LinkedIn.com)[4]

2.2 Random Forest

Random forest är en vanlig maskininlärningsalgoritm som kombinerar resultatet från flera beslutsträd för att nå ett enda resultat. Dess användarvänlighet och flexibilitet har underblåst dess antagande, eftersom det hanterar både klassificerings- och regressionsproblem.[5]



Figur 4. Random Forest: resultatet från flera beslutsträd för att nå ett enda resultat[5]

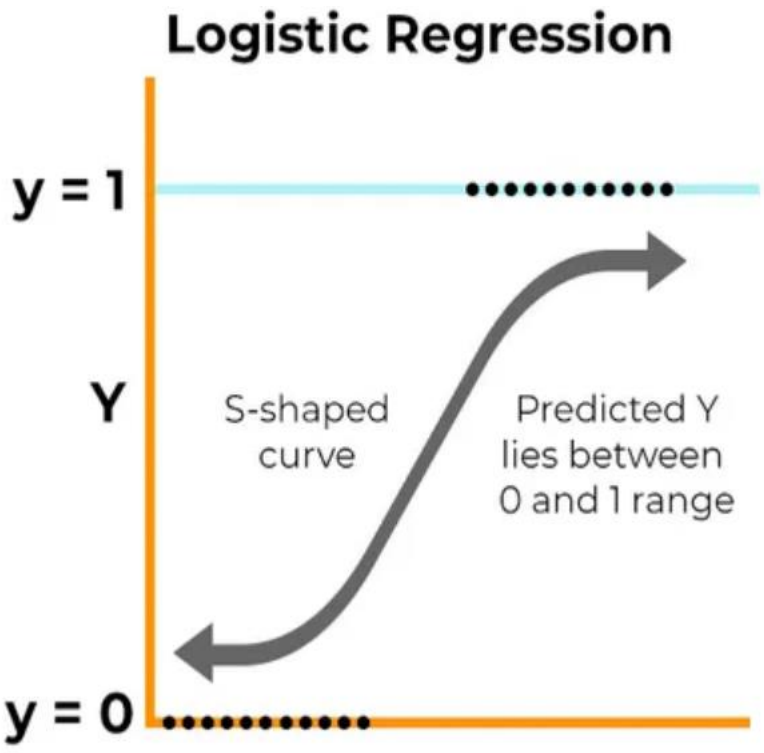
2.2.1 Hyperparameter för Random Forest

Slumpmässiga skogsalgoritmer har tre huvudhyperparametrar, som måste ställas in innan träning. Dessa inkluderar nodstorlek, antalet träd och antalet samplade funktioner. Därifrån kan den slumpmässiga skogsklassificeraren användas för att lösa regressions- eller klassificeringsproblem.[5]

(Källa: Random Forest på IBM.com)[5]

2.3 Logistic Regression

Logistisk regression uppskattar sannolikheten för att en händelse inträffar, till exempel röstade eller röstade inte, baserat på en given datamängd av oberoende variabler. Denna typ av statistisk modell används ofta för klassificering och prediktiv analys. Eftersom utfallet är en sannolikhet är den beroende variabeln avgränsad mellan 0 och 1.[6]



Figur 5. Logistic Regression[6]

2.3.1 Hyperparameter för Logistic Regression

De huvudsakliga hyperparametrarna vi kommer att ställa in i logistisk regression är solver, penalty, max_iter, C (regularization strength), tol, fit_intercept, intercept_scaling, class_weight, random_state, multi_class, verbose, warm_start och l1_ratio.[6]

(Källa: A Comprehensive Analysis of Hyperparameter Optimization in Logistic Regression Models på GitConnected.com)[6]

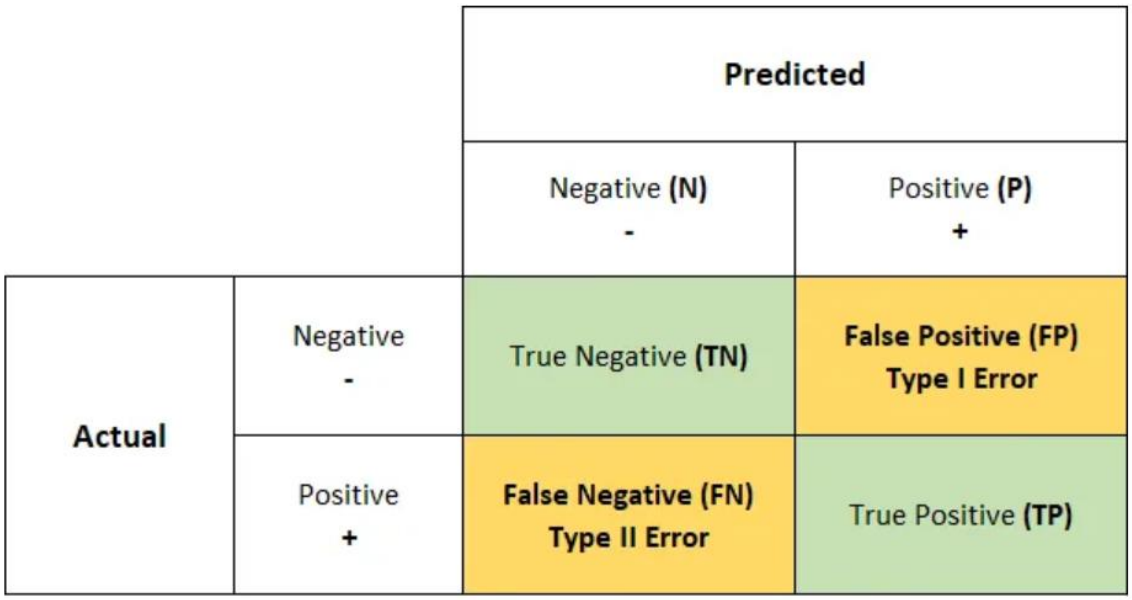
2.4 GridSearchCV (Grid Search Cross-Validation)

GridSearchCV är processen att utföra hyperparameterjustering för att bestämma de optimala värdena för en given modell. Det är en teknik för att hitta de optimala parametervärdena från en given uppsättning parametrar i ett rutnät. Det är i grunden en korsvalideringsteknik. Modellen samt parametrarna måste anges. Efter att ha extraherat de bästa parametervärdena görs förutsägelser.[7]

(Källa: Hyperparameter Tuning with GridSearchCV av Great Learning Team på GreatLearning.com)[7]

2.5 Confusion Matrix

En förvirringsmatris är en N x N-matris som används för att utvärdera prestandan hos en klassificeringsmodell, där N är antalet målklasser. Matrisen jämför de faktiska målvärdena med de som förutsägs av maskininlärningsmodellen.[8]



Figur 6. Confusion Matrix[8]

(Källa: "What is a confusion matrix?" av Anuganti Suresh på Medium.com)[8]

3. Metod

3.1 Konfigurera bibliotek

I början har vi konfigurerade bibliotek i Jupyter Notebook som behövs för att göra vårt projekt praktiskt. Vi använde främst Python-programmeringsspråket och scikit-learn-biblioteket.

Setup

In [102]:
```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
```

Figur 7. Konfigurera biblotek i Jupyter Notebook

3.2 Nedladdning av MNIST-datauppsättning

Efter att ha ställt in bibliotek som behövs för att göra detta projekt, laddade vi MNIST Dataset som är en berömd datauppsättning av handskrivna bilder med siffror som innehåller 70 000 bilder (vardera 28x28 pixlar) konverterade till en dimension.

MNIST Dataset

In [2]:
```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1, cache=True, as_frame=False)

X = mnist["data"]
y = mnist["target"].astype(np.uint8)
```

In [3]:
```
print(mnist.DESCR)
```

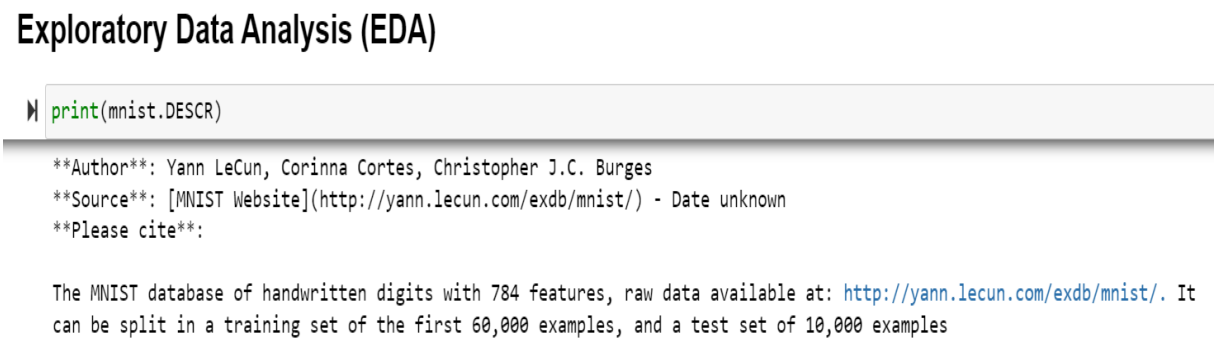
```
**Author**: Yann LeCun, Corinna Cortes, Christopher J.C. Burges
**Source**: [MNIST Website](http://yann.lecun.com/exdb/mnist/) - Date unknown
**Please cite**:

The MNIST database of handwritten digits with 784 features, raw data available at: http://yann.lecun.com/exdb/mnist/. It
can be split in a training set of the first 60,000 examples, and a test set of 10,000 examples
```

Figur 8. MNIST Dataset

3.3 Exploratory Data Analysis (EDA)

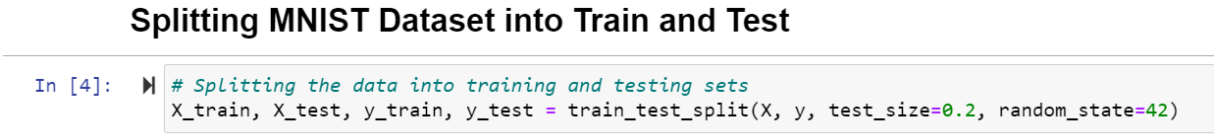
När MNIST-datan laddats gjorde vi lite utforskande dataanalys med dessa data som används för att analysera och undersöka datamängder och sammanfatta deras huvudsakliga egenskaper.



Figur 9. Exploratory Data Analysis (EDA)

3.4 Dela upp MNIST-datauppsättningen i Train och Test

I detta skede delade vi upp vår MNIST-datauppsättning i Train and Test. För tåg togs det 80% av data som är 56000 och för test, resten av data som är 14000.



Figur 10. Dela upp MNIST-datauppsättningen

3.5 Modell och hyperparameter

I det här skedet har vi provat tre olika modeller som är Support Vector Machine, Random Forest och Logistic Regression. Utöver det var den inställd hyperparameter.

It will be tried three different models below which are Support Vector Machine, Random Forest and Logistic Regression

```
# Iterating over each model and perform hyperparameter tuning
for model_name, param_grid in param_grids.items():
    if model_name == 'SVM':
        model = SVC(kernel='rbf')  # Use 'rbf' kernel
    elif model_name == 'RandomForest':
        model = RandomForestClassifier()
    elif model_name == 'LogisticRegression':
         model = LogisticRegression(solver='liblinear')  # Use 'liblinear' solver
```

Figur 11. Provar tre olika modeller

3.6 Hitta den bästa modellen

Slutligen hittade vi den bästa modellen av våra tre modeller bland Support Vector Machine, Random Forest och Logistic Regression.

Finding the best model out of the three models

```
# Getting the best model and its corresponding accuracy
best_model_candidate = grid_search.best_estimator_
y_pred = best_model_candidate.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

Figur 12. Hitta bästa modellen

4 Resultat och Diskussion

Vi har provat tre olika modeller och fann att Logistic Regression ansågs vara den bästa modellen bland dem med 92% noggrannhet. Efter det har vi provat 1000 värden genom denna modell för att testa och modellen gav resultat med 92% noggrannhet.

Predicitng some test values using the best model

```
# Predicting some test values using the best model and testing 1000 values
num_test_values = 1000
random_test_indices = np.random.choice(X_test.shape[0], size=num_test_values, replace=False)
X_test_sample = X_test[random_test_indices]
y_test_sample = y_test[random_test_indices]
y_pred_sample = best_model.predict(X_test_sample)
```

```
# Calculating accuracy of the predicted values
accuracy_sample = accuracy_score(y_test_sample, y_pred_sample)
print("Accuracy of predicted values:", accuracy_sample)
```

```
Accuracy of predicted values: 0.92
```

Figur 13. Bästa modellens förutsagda värden

Below is the comparison of true values and the best model's predicted values:

```
# Analyzing the accuracy of each prediction
for i in range(num_test_values):
    print(f"Predicted label: {y_pred_sample[i]}, True label: {y_test_sample[i]}, Result from the Model:
```

```
Predicted label: 1, True label: 1, Result from the Model: True
Predicted label: 5, True label: 5, Result from the Model: True
Predicted label: 7, True label: 7, Result from the Model: True
Predicted label: 0, True label: 0, Result from the Model: True
Predicted label: 3, True label: 3, Result from the Model: True
```

Figur 14. Jämförelse av modellförutsagda värden och sanna värden

5 Slutsatser

Vi har provat tre olika modeller för MNIST-datauppsättning som är Support Vector Machine, Random Forest och Logistic Regression. Den bästa modellen utvärderades Logistic Regression bland dem med 92% noggrannhet. Även om 92% är bra noggrannhet men inte perfekt. Om någon av modellerna skulle ge resultatet 95% noggrannhet eller mer så kommer det att vara en mycket bra modell. I så fall måste vi jobba mer i det här projektet och prova andra algoritmer, hyperparameter för att få bättre resultat än 92%.

6 Teoretiska frågor

1. Träningsdata är den data du använder för att träna en algoritm eller maskininlärningsmodell för att förutsäga resultatet av vår designade modell för att förutsäga.

Modellvalidering är den process som genomförs efter modellträning i maskininlärning där den tränade modellen utvärderas med en testdatauppsättning.

Testdata är en delmängd av träningsdata som används för att ge en objektiv utvärdering av en slutlig maskininlärningsmodell.

2. Eftersom Julia inte har någon explicit valideringsdatauppsättning, kan hon i det här fallet använda **korsvalideringsmetoden** för att jämföra prestandan för de tre modellerna och fatta ett beslut om vilken som passar bäst för hennes data. Cross-Validation är en metod för att testa prestandan hos en maskininlärningsmodell. Så genom att använda korsvalidering kan Julia göra en relativ jämförelse av de tre modellerna och välja den som ger bäst prestanda.

3. Regressionsproblem i maskininlärning är när en modell används för att förutsäga kontinuerliga resultat eller värden. Det är ett statistiskt tillvägagångssätt som används för att analysera sambandet mellan en beroende variabel (målvariabel) och en eller flera oberoende variabler (prediktorvariabler). Detta kan till exempel vara en modell som förutsäger löneförändringar, huspriser eller detaljhandel.

Några exempel på modeller:

1. Linear Regression
2. Polynomial Regression
3. Stepwise Regression
4. Decision Tree Regression
5. Random Forest Regression
6. Support Vector Regression
7. Ridge Regression
8. Lasso Regression
9. ElasticNet Regression
10. Bayesian Linear Regression

Potentiella tillämpningsområden:

1. Prediktiv modellering och prognoser
2. Marknadsföring och kundanalys
3. Finansiell analys
4. Sjukvård och medicinsk forskning
5. Kvalitetskontroll och processoptimering
6. Samhällsvetenskap och beteendeanalys
7. Energi och verktyg
8. Sportanalys
9. Miljöanalys
10. Tidsserieanalys

4. Root Mean Square Error, RMSE=n1∑i=1n(yi−yi^)2 är en metod som används för att utvärdera riktigheten av en regressionsmodells förutsägelser. Den mäter den genomsnittliga avvikelsen för de förutsagda värdena från de faktiska värdena. Med andra ord, det är standardavvikelsen för residualerna som representerar avståndet mellan regressionslinjen och datapunkterna.

För utvärdering av mått för regressionsmodeller, särskilt när felens skala är viktig.
Förstå hur bra modellen presterar när det gäller prediktionsnoggrannhet.
Jämför olika modeller eller ställ in hyperparametrar för att hitta den modell som ger bäst prestanda.
Bedöma om ytterligare förbättringar behövs för modellen eller om den är tillfredsställande för implementering.

5. Klassificeringsproblem i maskininlärning är ett prediktivt modelleringsproblem där klassetiketten förutses (målet är att kategorisera objekt) för ett specifikt exempel på indata.

De vanligaste klassificeringsmodellerna är nämligen Logistic Regression, Support Vector Machines (SVM), Decision Trees and Random Forests, K-Nearest Neighbors (KNN) och Naive Bayes.

Några exempel på potentiella användningsområden för ovan nämnda modeller: filtrering av skräppost för e-post, medicinskt diagnostiskt test, bedrägeriupptäckt, bildigenkänning, ansiktsigenkänning, röstigenkänning, språkidentifiering och handskriftsigenkänning.

En confusion matrix är ett prestationsutvärderingsverktyg inom maskininlärning, som representerar noggrannheten hos en klassificeringsmodell. Den visar antalet sanna positiva, sanna negativa, falska positiva och falska negativa.

6. K-means modellen är en unsupervised maskininlärning-algoritm som är en metod för att gruppera n observationer i K-kluster. Den använder vektorkvantisering och syftar till att tilldela varje observation till klustret med närmaste medelvärde eller tyngdpunkt, som fungerar som en prototyp för klustret.

Användningsområde för K-means-modellen:
Kundsegmentering, dokumentkluster, bildsegmentering, bildkomprimering, bedrägeriupptäckt, identifiering av cyberbrott och optimering av leveransvägar.

7. Ordinal encoding fungerar genom att mappa varje unikt kategorivärde till ett annat heltal. Vanligtvis börjar heltal vid 0 och ökar med 1 för varje ytterligare kategori. Till exempel skulle en "storlek"-variabel med värden ["small", "medium", "large"] mappas till [0, 1, 2].

One-hot encoding är en metod som vanligtvis används för att hantera kategoriska data i maskininlärning. Det är en process som används för att konvertera kategoriska datavariabler till en form som kan tillhandahållas till maskininlärningsalgoritmer för att göra ett bättre jobb i förutsägelse.

Exempel:
One-hot encoding skapar nya (binära) kolumner, som indikerar närvaron av varje möjligt värde från originaldata som kan ses nedan:

Color
Red
Blue
Green

Efter One-hot encoding där den konverterades k-variabler från k olika kategorier:

Red	Blue	Green
1	0	0
0	1	0
0	0	1

Dummy variable encoding som är kategoriska datakodningsmetod omvandlar den kategoriska variabeln till en uppsättning binära variabler. Dummy-kodning lägger till färre dummy-variabler än en-hot-kodning gör.

Till exempel, om vi har en kategorisk variabel som har k olika kategorier kommer en-hot-kodning att konvertera den till k-variabler medan dummy-kodning kommer att konvertera den till k-1-variabler.

Color
Red
Blue
Gree

Efter Dummy variable encoding där den konverterades k-1-variabler från k olika kategorier:

Red	Blue
1	0
0	1
0	0

8. Vi vet att data i stora drag kan kategoriseras i två huvudtyper: **ordinal och nominal.**

Nominal data: denna typ av data består av kategorier som inte har någon ordning eller rangordning. Till exempel färger som är {röd, grön, blå}.

Ordinal data: denna typ av data har en naturlig ordning eller rangordning mellan kategorier, till exempel rankningar som är {1:a plats, 2:a plats, 3:e plats}.

Nu kommer vi till punkten vad Göran och Julia har berättat:

Göran berättade att "data är antingen ordinal eller nominal". Vi kan se av vår definition ovan inklusive exempel att det är sant och Göran har rätt.

Julia berättade att "färger som röd, grön och blå är nominal data men en röd skjorta är det inte". Här introducerar hon en subjektiv rangordning som är röd skjorta. Så, röd färg är en typ av nominal data medan röd färg skjorta en typ av ordinal data. Enligt definitionen och exemplet har Julia också rätt.

Därför har både Göran och Julia rätt beroende på sammanhanget. Vi kan säga, Göran angav klassificeringen av datatyp och Julia gjorde förklaring på Görans uttalande.

9. Streamlit är ett Python-bibliotek med öppen källkod som gör det enkelt att skapa och dela vackra, anpassade webbappar för maskininlärning och datavetenskap. Genom Streamlit är det möjligt att bygga och distribuera kraftfulla dataappar på några minuter.

Användarvänlighet:
Streamlit är det enklaste sättet, särskilt för personer utan front-end-kunskaper att lägga sin kod i en webbapplikation där Ingen front-end (html, js, css) erfarenhet eller kunskap krävs.

Datavisualisering: Streamlit ger inbyggt stöd för datavisualiseringsbibliotek som Matplotlib, Plotly och Altair.

Prototyper och snabb utveckling: Streamlit är utmärkt för prototyper och snabbt utvecklande webbapplikationer.

Python-integration: Streamlit är ett naturligt val eftersom det är skrivet i Python.

Gemenskap och ekosystem: Streamlit har en växande community och ett ekosystem av tillägg och anpassade komponenter som kan förbättra dess funktionalitet.

7 Självutvärdering

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

Det tog lång tid när jag försökte köra koden för att ställa in hyperparameter. Jag fick flera varningar också. Sedan försökte jag med olika hyperparameter för olika modeller respektive och till slut var problemet löst.

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

Jag hoppas på att få godkänt betyg denna gång eftersom jag inte klarade av VG del. Jag känner mig väldigt ledsen för detta eftersom VG del är väldigt intressant för mig men kunde inte göra det på grund av vissa personliga problem.

3. Något du vill lyfta fram till Antonio?

Nej.

Skulle gärna vilja nämna här att han är en av de bästa lärarna i mitt liv. Därför att jag skrev hans namn som kurslärare på första sidan i denna rapport.

Jag önskar honom all lycka för hans framtida yrkeskarriär!

Källförteckning

1. https://www.linkedin.com/pulse/deep-learning-action-building-training-neural-network-rany-yvuic/

2. https://www.linkedin.com/pulse/mnist-dataset-classification-atif-ahmed/

3. https://www.ibm.com/topics/support-vector-machine

4. https://www.linkedin.com/advice/0/how-do-you-optimize-hyperparameters-svm#:~:text=SVM%20hyperparameters%20are%20the%20settings,decision%20boundary%20can%20be%20found.

5. https://www.ibm.com/topics/random-forest

6. https://levelup.gitconnected.com/a-comprehensive-analysis-of-hyperparameter-optimization-in-logistic-regression-models-521564c1bfc0

7. https://www.mygreatlearning.com/blog/gridsearchcv/

8. https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5