Maskininlärning för sifferigenkänning



Arina Godman

EC Utbildning

202403

# Abstract

This project focuses on digit recognition utilizing a machine learning algorithm that’s is chosen from models such as: Support Vector Classifier, RandomForest, GradientBoostingClassifier, and a HardVoting Classifier composed of those models. The objective is to select the most effective model and develop an application for handwritten digit recognition.

The models are trained on a subset of the MNIST dataset, that in total contains 70,000 images of handwritten digits and their corresponding labels. Model performance is evaluated using techniques such as the Confusion Matrix and accuracy score.

Upon identifying the optimal model, it is trained on the entire MNIST-data set and implemented into a Streamlit application for digit recognition, offering users an interactive platform for precise digit identification.

Innehållsförteckning

[Abstract 2](#_Toc161925582)

[1 Inledning 2](#_Toc161925583)

[2 Teori 3](#_Toc161925584)

[2.1 Klassificeringsmodeller 3](#_Toc161925585)

[2.1.1 Support Vector Classifier 3](#_Toc161925586)

[2.1.2 Random Forest Classifier 3](#_Toc161925587)

[2.1.3 Gradient Boosting Classifier 4](#_Toc161925588)

[2.1.4 Voting Classifier 4](#_Toc161925589)

[2.2 Modellutvärdering och modellval 4](#_Toc161925590)

[2.2.1 Tränings-, validerings- och testdata 4](#_Toc161925591)

[2.2.2 Val av hyperparametrar 4](#_Toc161925592)

[2.2.3 Utvärdering av klassificeringsmodeller med Confusion Matrix 4](#_Toc161925593)

[2.3 Streamlit 5](#_Toc161925594)

[3 Metod 6](#_Toc161925595)

[3.1 Verktyg 6](#_Toc161925596)

[3.2 Data 6](#_Toc161925597)

[3.2.1 Utforskning av träningsdatan 6](#_Toc161925598)

[3.2.2 Datapreparation 6](#_Toc161925599)

[3.3 Modelval 7](#_Toc161925600)

[3.3.1 Utvärdering av modeller 7](#_Toc161925601)

[3.4 Applikation 7](#_Toc161925602)

[3.4.1 Modell 7](#_Toc161925603)

[3.4.2 Uppbyggnad 7](#_Toc161925604)

[3.4.3 Utseende av appen och funkionalitet 7](#_Toc161925605)

[3.4.4 Förberedning av indata 8](#_Toc161925606)

[4 Resultat och Diskussion 9](#_Toc161925607)

[4.1 Modellval 9](#_Toc161925608)

[4.2 Applikation 10](#_Toc161925609)

[4.2.1 Uppladdade bilder 10](#_Toc161925610)

[4.2.2 Canvas-bilder 10](#_Toc161925611)

[5 Slutsatser 12](#_Toc161925612)

[6 Teoretiska frågor 13](#_Toc161925613)

[7 Självutvärdering 15](#_Toc161925614)

[Källförteckning 16](#_Toc161925615)

# Inledning

Handskriven sifferigenkänning är av stor relevans i dagens teknologiska framsteg på grund av dess mångsidiga tillämpningar, inklusive automatiserade sorteringsystem, digitalisering av handskrivna dokument och automatisk identifiering av handskrivna texter. Dess potential att effektivisera processer och förbättra användarupplevelsen understryker dess betydelse inom olika sektorer, vilket gör det till en central punkt för forskning och utveckling inom maskininlärning.

Syftet med denna rapport är att utforska och jämföra olika maskininlärningsmodeller för sifferigenkänning med målet att välja den mest effektiva modellen för att utveckla en applikation för igenkänning av handskrivna siffror. För att uppfylla syftet kommer följande frågeställningar att besvaras:

1. Vilken av de undersökta maskininlärningsmodellerna presterar bäst för sifferigenkänning baserat på träning på MNIST-datasetet?
2. Hur kan den valda modellen implementeras i en Streamlit-applikation för att möjliggöra interaktiv igenkänning av handskrivna siffror?
3. Vilka begränsningar har den valda modellen?

# Teori

## Klassificeringsmodeller

Klassificering i maskininlärning är en central metod för att kategorisera data i olika grupper eller klasser baserat på dess egenskaper eller attribut. Genom att utnyttja olika algoritmer, som exempelvis beslutsträd, support vector machines eller neurala nätverk, kan maskininlärningsmodeller tränas för att utföra klassificeringsuppgifter med hög precision och generaliseringsförmåga.

### Support Vector Classifier

Support Vector Classifier (SVC) är en maskininlärningsalgoritm som används för att lösa klassificeringsproblem genom att hitta den optimala hyperplanet som kan separera datapunkterna i olika klasser med maximal marginal (Géron, A., 2019, p.153). Genom att identifiera så kallade support vectors, datapunkter närmast hyperplanet, försöker SVC maximera avståndet mellan dem för de olika klasserna. Algoritmen arbetar med att först transformera data till ett högre dimensionsutrymme där klasserna kan separeras linjärt, med hjälp av en kärnfunktion (Géron, A., 2019, p.154). Sedan tränas modellen genom att optimera en objektivfunktion som syftar till att minimera både klassificeringsfelet och komplexiteten hos den resulterande hyperplanet, vilket görs med stöd av tekniker som stödvektormaskiner (Géron, A., 2019, p.155). När modellen är tränad, kan den användas för att klassificera nya datapunkter genom att avgöra vilken sida av hyperplanet de befinner sig på.

#### Några hyperparametrar

Parametern C (Regularization) styr överjusteringen. Ett mindre C-värde leder till en mjukare beslutningsgräns som tillåter vissa felklassificeringar i träningsdata, medan ett högre C-värde strävar efter att korrekt klassificera alla exempel och kan leda till överjustering.  
Kernel anger vilken kärnfunktion som ska användas i algoritmen. Vanliga kärnfunktioner inkluderar 'linear', 'poly' (polynomial), och 'rbf' (radial basis function).  
Gamma är en parameter för 'rbf', ’poly’ och ’sigmoid’-kärnor och styr inverkan av varje träningspunkt. Ett högre gamma-värde innebär att en punkt måste vara nära för att påverka klassificeringen, medan ett lägre gamma-värde betyder att punkter längre bort också kan påverka klassificeringen.

#### Standard Scaler

För att optimera prestandan hos SVC kan det vara nödvändigt att standardisera parametrarna. Att standardisera parametrar innebär att man subtraherar medelvärdet och dividerar med standardavvikelsen (Géron, 2019, s. 59). Detta är särskilt viktigt när egenskaperna i datamängden har olika skalor eller enheter och kan leda till en mer balanserad och stabil modell.

### Random Forest Classifier

RandomForest Classifier är en populär algoritm inom maskininlärning som tillhör kategorin ensemble-metoder. Den bygger på principen att kombinera flera beslutsträd för att öka prediktionskraften och stabiliteten hos modellen. Genom att slumpmässigt välja olika delmängder av träningsdata för varje träd och sedan kombinera deras resultat, kan RandomForest Classifier minska risken för överanpassning och ge mer pålitliga förutsägelser (Breiman, 2001, s. 5).

#### Några hyperprametrar av Random Forest Classifier

Hyperparametern n\_estimators specificerar antalet träd i skogen. Att öka detta antal kan öka modellens prestanda, men det leder också till högre beräkningskostnader. Max\_depth bestämmer det maximala djupet för varje träd i skogen. Ett högre max\_depth kan resultera i överanpassning, medan ett lägre värde kan minska modellens komplexitet och förbättra dess generaliseringsförmåga. Min\_samples\_split anger det minsta antalet exempel som krävs för att dela en inre nod. Att öka detta värde kan leda till en enklare modell och minska risken för överanpassning. Min\_samples\_leaf specificerar det minsta antalet exempel som krävs för att en nod ska vara en lövnod. Att öka detta värde kan bidra till att skapa en mer robust modell och minska risken för överanpassning.

### Gradient Boosting Classifier

GradientBoostingClassifier är en annan populär ensemble-modell. Algoritmen fungerar genom att successivt konstruera flera svaga modeller, vanligtvis beslutsträd, där varje träd korrigerar felet från den föregående modellen (Géron, A., 2019, p.203). Genom att fokusera på att minimera gradienten av förlustfunktionen kan GradientBoostingClassifier ge hög prediktionskraft och generaliseringsförmåga, även för komplexa datamängder. En fördel med denna algoritm är dess förmåga att hantera både klassificerings- och regressionsuppgifter och dess robusthet mot överanpassning (Géron, A., 2019, p.203).

### Voting Classifier

Voting Classifier är en metod inom ensemble learning, som kombinerar flera enskilda maskininlärningsmodeller för att förbättra prestandan hos den resulterande modellen. I denna metod tränas flera olika klassificerare, var och en på samma träningsdata. Sedan kombineras deras individuella förutsägelser för att skapa en samlad förutsägelse genom majoritetsröstning (hard voting) eller genomsnittlig avröstning (soft voting) (Géron, A., 2019, p.190). Genom att använda flera olika klassificerare som använder olika tillvägagångssätt kan Voting Classifier ge bättre generalisering och prestanda jämfört med enskilda klassificerare.

## Modellutvärdering och modellval

I detta avsnitt utforskar vi olika tekniker för att identifiera den mest effektiva modellen och optimera dess prestanda genom val av hyperparametrar. Genom att noggrant utvärdera och välja rätt modell kan vi säkerställa att våra maskininlärningsprojekt uppnår önskad noggrannhet och generaliseringsförmåga. Processen brukar börja med att man tar några olika modeller, som man tränar upp på en del av tillgängliga data, utvärderar och väljer den bästa modellen som man kommer att använda i sitt projekt.

### Tränings-, validerings- och testdata

En av viktiga metoder för att säkerställa en korrekt utvärdering och generalisering av modeller är att dela datamängden i tränings-, test- och valideringsset. Träningssetet används för att träna modellen och anpassa dess parametrar, medan testsetet fungerar som en oberoende uppsättning för att objektivt utvärdera modellens prestanda. Valideringssetet är avgörande för finjustering av modellen och välja de bästa hyperparametrarna för att undvika överanpassning. Genom att använda separata set kan man förhindra att modellen överanpassas till träningsdata och säkerställa att den kan generalisera väl till nya, oberoende data.

### Val av hyperparametrar

När man väljer hyperparametrar i maskininlärningsmodeller är det viktigt att utforska olika kombinationer för att hitta den konfiguration som ger bäst prestanda. GridSearchCV är en populär teknik för detta, där man systematiskt testar olika hyperparametrar genom att skapa en "grid" av möjliga värden. Genom att använda korsvalidering kan GridSearchCV utvärdera varje kombination av hyperparametrar och identifiera den som ger bäst resultat (Géron, A., 2019, p.76). På detta sätt kan man optimera modellens prestanda och generaliseringsförmåga över olika datamängder

### Utvärdering av klassificeringsmodeller med Confusion Matrix

Att utvärdera prestandan hos en klassificerare kan förbättras avsevärt genom att använda en förvirringsmatris (Confusion Matrix), där antalet gånger instanser av en viss klass klassificeras som en annan klass räknas. I förvirringsmatrisen är de sanna klasserna av observationerna längs den vertikala axeln och de förutsagda klasserna längs den horisontella axeln, vilket gör att alla korrekta prediktioner hamnar på huvuddiagonalen. Vid multiklassklassificering, när det finns mer än två klasser, är förvirringsmatrisen mycket användbar för att snabbt utvärdera hur modellen presterar för varje enskild klass. Dessutom används förvirringsmatris som utgångspunkt i beräkning av viktiga värden som precision, recall, f1 score och accuracy. Dessa metriker beräknas genom att på olika sätt jämföra de korrekt klassificerade instanserna av positiva och negativa klasser, kända som TP (True Positive) och TN (True Negative), med motsvarande FP (False Positive) och FN (False Negative) för felaktigt klassificerade instanser. För multiklass klassificering beräknas dessa metriker på förljande sätt:

#### Precision

Precision representerar förmågan hos klassificeraren att inte ge falska positiva för en viss klass. En hög precision indikerar att få instanser av andra klasser har klassificerats som den aktuella klassen.

#### Recall

Recall representerar förmågan hos klassificeraren att hitta alla sanna positiva för en viss klass. En hög recall indikerar att klassificeraren kan identifiera de flesta av de sanna instanserna av den aktuella klassen.

#### F1-score

F1-score är ett balanserat mått som kombinerar både precision och recall. Den tar hänsyn till både antalet falska positiva och falska negativa. Ett högt F1-score indikerar både hög precision och hög recall, vilket är önskvärt för en klassificerare.

#### Accuracy

Accuracy representerar andelen korrekt klassificerade instanser av alla instanser. Det är ett generellt mått på modellens övergripande prestanda. En hög accuracy indikerar att klassificeraren lyckas korrekt klassificera de flesta instanser av alla klasser.

## Streamlit

Streamlit är en populär plattform för att bygga och distribuera interaktiva webbapplikationer för maskininlärning och dataanalys. Med dess enkla och intuitiva gränssnitt kan användare skapa kraftfulla applikationer utan att behöva djupa kunskaper inom webbutveckling. Streamlit erbjuder även en mängd olika funktioner och verktyg för att underlätta utvecklingsprocessen och maximera användarupplevelsen. Enligt Streamlits officiella webbplats erbjuder plattformen "en enkel och snabb väg till produktion" för datavetare och utvecklare, vilket gör den till ett populärt val för att snabbt omvandla modeller till användbara applikationer (Streamlit, 2024.)

# Metod

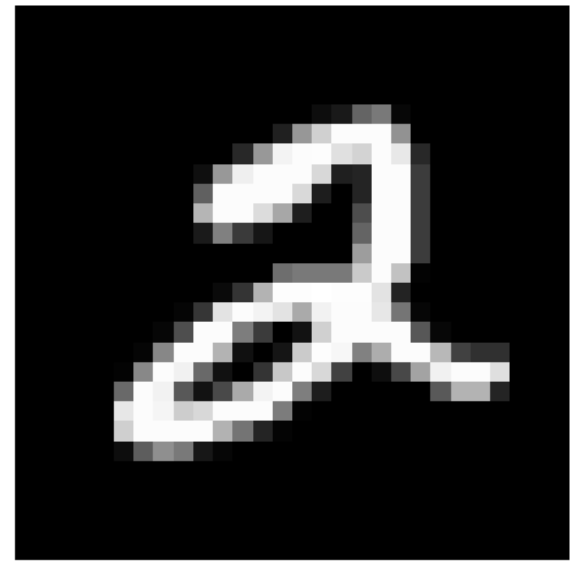
I detta kapitel behandlas ursprunget till och förberedelserna som görs för datan, modellval och val av begränsningar, hur vi utvärderar modellprestanda och förklarar de genomförda experimenten.

## Verktyg

För datanalys, modellval och träningsprocessen användes Python (bibliotek som numpy, sci-kit learn och joblib) i Jupyter Notebook. Streamlit-applikationen utvecklades med Python i Visual Studio Code (VS Code) på grund av dess fördelar med att hantera flera projektfiler samtidigt och möjligheten att enkelt ladda upp dem till GitHub direkt från VS Code med hjälp av GitHub-tillägget.

## Data

För detta projekt användes en välkänd MNIST-dataset som hämtades med hjälp av funktionen fetch\_openml() från biblioteket sklearn.datasets. Datasetet består av 70 000 bilder med handskrivna vita siffror från 0 till 9 på en svart bakgrund. Datan delades upp i tre delar: träningssetet innehållande 10 000 bilder, valideringssetet med 2 000 bilder och testsetet med 2 000 bilder. Denna uppdelning var nödvändig för att kunna välja den högst presterande modell bland några utvlda och gjordes med hjälp av funktion traintestsplit() från bibliotek sklearn.preprocessing



Figur 1. Exempelbild från MNIST (Siffra 2)

### Utforskning av träningsdatan

För att undvika omedveten bias utförde jag analys på träningsdatan och inte hela datasetet. Varje observation visade sig ha 784 attribut (pixlar) vilket innebär att varje bild har format 28x28 pixlar. Träningssetet visade sig inte innehålla saknade värden. Fördelningen av observationer mellan olika klasser (siffror) i datasetet var följande:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1001 | 1127 | 991 | 1032 | 980 | 863 | 1014 | 1070 | 944 | 987 |

Tabell 1. Fördelning av observationer i tränigsset

### Datapreparation

Eftersom datan var redan av bra kvalite den enda preparation som behövdes var standardiseringen av datan med StandardScaler från sklearn. Detta var nödvändig, särskilt eftersom en av modellerna var en SVC (Support Vector Classifier). SVC-modellen är känslig för skalning av data, och genom att tillämpa standardisering kunde vi säkerställa att modellen behandlade attributen på ett enhetligt sätt och därigenom förbättra dess prestanda.

## Modelval

För att hitta de mest passande parametrarna för SVC och RandomForest utfördes en sökning bland olika hyperparametervärden med hjälp av GridSearchCV från sklearn.model\_selection. Genom att använda GridSearchCV kunde vi systematiskt testa olika kombinationer av hyperparametrar och identifiera de som gav bäst prestanda för varje modell.

### Utvärdering av modeller

För att utvärdera modellerna använde jag ConfusionMatrix och accuracy\_score från sklearn.metrics. Den slutliga modellen valdes baserat på det högsta accuracy\_score samt en ConfusionMatrix som inte hade extrema felaktiga värden. Genom att använda dessa metriker kunde vi bedöma modellernas prestanda och välja den mest lämpliga för vårt syfte.

## Applikation

Efter att ha valt modellen byggde jag en Streamlit-applikation för att möjliggöra sifferigenkänning och för att dela den med en bredare användargrupp. Med Streamlit kunde jag skapa ett användarvänligt gränssnitt som gör det enkelt för användare att interagera med modellen och utföra sifferigenkänning utan behov av avancerad programmeringskunskap. Detta öppnar upp möjligheten att dela och använda modellen på ett mer tillgängligt och användarvänligt sätt.

### Modell

Efter att modellen valdes har jag återtränat den på hela datasetet och laddat ner den till en separat fil med hjälp av biblioteket joblib. Samma procedur genomfördes med StandardScaler som behövs för att standardisera input-bilder. Därefter laddades modellen och StandardScaler upp i min applikation för användning.

### Uppbyggnad

Applikationen består av två Python-filer: app.py och prediction.py. I app.py definierar jag utseendet på min Streamlit-app samt grundläggande preprocessing av input-bilder. I filen prediction.py laddar jag upp modellerna och definierar funktionerna preprocess\_image() och make\_prediction(). preprocess\_image() används för att standardisera bilden så att den får rätt formatering för att modellen ska kunna göra korrekta prediktioner, medan make\_prediction() är en funktion som tar den processerade bilden som input och returnerar dess prediktion, en siffra.

### Utseende av appen och funkionalitet

I min app prioriterade jag funktionalitet över utseende. Användare kan med hjälp av en kryssruta välja om de vill ladda upp sin bild från sin dator eller rita den på en canvas. Som standard kan användare endast ladda upp en JPG, JPEG eller PNG-fil som är mindre än 200 MB. I detta fall visas den uppladdade bilden på skärmen. Om användaren istället väljer att rita på en canvas, behöver den markera kryssrutan vilket gör att uppladdningsblocket försvinner och en canvas visas där de kan rita sina bilder. Under uppladdningsblocket och canvasen finns knappen "Predict", där användaren kan trycka för att få förutsägelser. Förutsägelserna visas som text under canvasen eller bilden. Du kan hitta screen-shots av appen på nästa sida.

En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivningEn bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

Figur 2. Applikation

### Förberedning av indata

Data från MNIST-datasetet, som bestod av bilder av handskrivna siffror, var varierande men inte direkt anpassade till verkliga förhållanden. För att modellen skulle kunna göra korrekta prediktioner var det nödvändigt att anpassa de inkommande bilderna. För detta ändamål använde jag biblioteket cv2.

Som första steg i förberedelserna konverterade jag bilderna till gråskala och ändrade deras storlek till 28 x 28 pixlar med hjälp av biblioteket cv2. Därefter behövde vi ändra färgen på både bakgrunden och själva siffrorna, eftersom MNIST-bilderna presenterar vita siffror på en svart bakgrund medan uppladdade bilder vanligtvis är svarta siffror på en vit bakgrund. För att göra detta utförde jag en inversion genom att subtrahera pixelvärdet från 255. Detta steg var endast nödvändigt för uppladdade bilder, eftersom vår canvas-alternativ redan genererar vita bilder på en svart bakgrund.

För det tredje, eftersom siffrorna i MNIST-datasetet hade helt svart bakgrund var det nödvändigt att våra bilder före inversion hade en helt vit bakgrund. Detta var dock en utmaning eftersom det är svårt att uppnå en helt vit bakgrund i verkliga bilder, och det kunde även finnas skuggor på bilderna som kunde påverka modellens prestanda. För att rätta detta och uppnå en helt svart backgrund på våra bilder införde jag en tröskel som färgade om alla pixlar som var lika med eller lägre än träningsdatans medelvärde till svart. Även detta steg behövdes inte för canvas-bilder.

Som sista steg blev bildens data plattad och omformad för att sedan standardiseras med StandardScaler. Därefter var den redo att skickas till modellen för prediktion.

# Resultat och Diskussion

I det här kapitlet vill jag presentera resultat av modellval bland 4 modeller: SVC, RandomForest, GradientBoostingClassifier och VotingClassifier och implementering av den utvalda modellen i min Streamlit applikation.

## Modellval

För att välja de optimala hyperparametrarna som skulle leda till den mest presterande modellen genomförde jag gridsearch för två modeller med hjälp av GridSearchCV och trainset.  
Bästa parametrar visade sig med accuracy på 96.06% vara {'C': 0.5, 'gamma': 1, 'kernel': 'poly'} och för RandomForestClassifier med accuracy på 95.65% {'max\_depth': None, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 200}.

Nere kan du se en tabell med prestation av alla modeller på en valideringsset och ConfusionMatrices.

|  |  |  |  |
| --- | --- | --- | --- |
| Accuracy för olika modeller | | | |
| SVC | RandomForestClassifier | GradientBoostingClassifier | VotingClassifier |
| 96,06% | 95,65% | 93,25% | 95,9% |

|  |  |
| --- | --- |
| En bild som visar skärmbild, kvadrat, Färggrann, text  Automatiskt genererad beskrivning  Figur 3. ConfusionMatrix SVC. | En bild som visar skärmbild, Färggrann, kvadrat, mönster  Automatiskt genererad beskrivning  Figur 4. ConfusionMatrix - RandomForestClassifier |
| En bild som visar skärmbild, kvadrat, Färggrann, mönster  Automatiskt genererad beskrivning  Figur 5. ConfusionMatrix GradientBoostingClassifier | En bild som visar skärmbild, Färggrann, kvadrat, mönster  Automatiskt genererad beskrivning  Figur 6. ConfusionMatrix VotingClassifier |

Tabell 2. Accuracy för olika modeller

Baserat på accuracy uppnådde SVC och Voting Classifier de högsta värdena med 96.96% och 95.9% respektive. Efter att ha jämfört deras ConfusionMatrices valde jag att gå vidare med SVC. Även om båda modellerna hade liknande ConfusionMatrices, visade SVC bättre resultat för klasser 3, 6, 8 och 9. Dessutom krävde träningen av en SVC-modell mindre tid och energi jämfört med Voting Classifier.  
  
Efter att jag valde SVC ville jag vara extra säker på att det är en modell som kan prestera bra. Jag tränade därför om den på både tränings- och valideringsdata och testade sedan på testdatan. När jag fick en noggrannhet på 96.25% blev jag övertygad om att den höga noggrannheten på valideringsdatan inte var en slump.

## Applikation

Efter att jag har implementerat modellen i applikationen och distribuerat appen var det dags att testa hur modellen skulle prestera med riktiga data från användarna.

### Uppladdade bilder

Modellen har predikterat korrekta värden för de uppladdade siffror och även dem siffrorna som var inte så noggrant skrivna. Till exemplen de har olika mått, bakgrundsfärg och lutning. Se exempel på bilden nedan.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Figur 7. Exempel på fur modellen fungerar med bilduppladdning

Det är viktigt att notera att jag medvetet valde en tjockare penna för att rita siffrorna så att de skulle likna siffrorna på MNIST-bilderna. Vid tester med en mycket tunnare penna kunde inte rätt prediktioner erhållas. Dessutom var det nödvändigt att centrera varje siffra för att få korrekta prediktioner.

### Canvas-bilder

Modellen kunde också korrekt förutsäga klasserna för bilder som ritades på en canvas, åtminstone en gång för varje klass. Trots att jag lagt till information i applikationen om att bilderna behöver vara centrerade, hade modellen svårigheter att göra korrekta prediktioner för varje ritad siffra från första försöket. Jag utförde även tester där jag ändrade pennans tjocklek, men det gav inte mycket förbättring. Jag antar att skillnaderna i hur bilder ritas på papper jämfört med en pekplatta på en bärbar dator eller med en datormus kan vara en faktor. Eftersom modellen var tränad på handskrivna bilder kan problemet ligga just där. Exempel på fungerande canvas-lösning hitar du på nästa sida.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Figur 8. Exempel på prediktioner av canvas-lösningen

# Slutsatser

Enligt min undersökning av olika maskininlärningsalgoritmer visar resultaten att SVC (Support Vector Classifier) presterar bäst för sifferigenkänning jämfört med Random Forest Classifier, Gradient Boosting Classifier och Voting Classifier. Studierna visade också att det är möjligt att skapa en fungerande Streamlit-applikation och implementera modellen på ett sätt som möjliggör korrekta prediktioner. Detta indikerar att SVC kan vara ett föredraget val för sifferigenkänning i praktiska tillämpningar.

Däremot visade det sig att SVC-modellen, tränad på MNIST-data, hade vissa begränsningar. För det första måste siffrorna på bilderna vara centrerade för att modellen ska göra korrekta prediktioner. Dessutom är det viktigt att använda en viss pennstorlek när man ritar siffrorna och att siffrorna är helt handskrivna för att få bästa resultat. Och för det tredje är att siffrorna ska ritas med hand, inte på en pekskärm eller med datormus.   
  
Denna modell kan vara användbar för att tolka siffror i en produktionsmiljö eller annan kontrollerad miljö där vi kan kontrollera att siffrorna ska skrivas med tillhandahållna medel (som vi godkänner) och där vi också kan ha kontroll över hur bilderna tas på dessa siffror (utan lutning och skuggor).

För att göra denna modell användbar för en bredare användargrupp eller för underhållning som en Streamlit-app anser jag att det krävs något mer kraftfullt än en traditionell maskininlärningsmodell, nämligen neurala nätverk. Neurala nätverk har förmågan att lära sig mer komplexa mönster och kan vara mer flexibla när det gäller att tolka olika skrivstilar och variationer i handskrivna siffror. Genom att implementera neurala nätverk kan vi förbättra modellens förmåga att göra korrekta prediktioner och öka dess användbarhet för en bredare publik.

# Teoretiska frågor

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

Träning-set används för att träna upp olika modeller som sedan ska utvärderas på validerings-set. Sedan väljer man den bästa modellen, tränar upp den på träning+validering set och utvärderar den på Test-set.

**2. 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 kan använda K-fold cross-validation för att mäta modellernas prestanda utan att röra test-set. K-fold Cross-validation innebär att man delar upp träningsdata i några (K) olika delar, tränar upp modellen på K – 1 delar och validerar på den återstående delen. Man gör så K gånger för att få alla möjliga kombinationer av träning-validering delar av data och väljer den bästa modellen baserat på snitt-prestanda av de K-mätningar som man får från K-Fold cross-validation.

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

Regressionsproblem innebär att det behövs prediktion av en kontinuerlig målvariabel (den som inte har bestämda värden). Exempel på modeller: LinearRegression (och Lasso, Ridge), SupportVectorRegression (SVR), DesicionTreeRegressor.   
Potentiella tillämpningsområden: temperature-prediction (väderprognos), prediktion av huspriser, energiförbrukning, prediktion av när en buss kommer till en busstation.

**4. Hur kan du tolka RMSE och vad används det till?**

RMSE är roten ur kvadrerad skillnad mellan faktiska målvariabelsvärden och dess prediktioner i snitt. Den mäter en snitt skillnad mellan de faktiska värden av y och dess predikterade värden i samma enheter som y mäts i.

Det används till att mäta modellens prestanda. Ju mindre RMSE är, desto bättre presterar modellen.

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

Klassificeringsproblem innebär att uppgiften handlar om att prediktera om en observation tillhör någon av förbestämda klasser.

Exempel på modeler: LogistiskRegression, SupportVectorClassifier(SVC), DesicionTreeClassifier, K-nearest Neighbors (KNN)

Tillämpningsområden: Spamfiltrering, anomalidetektion, prediktera om kunden kommer att tycka om låten baserad på musik som den tidigare har lyssnat på (Spotify-wrap med deras hälsningar från musiker).

En Confusion Matrix är ett verktyg för att utvärdera klassificeringsmodeller. Det är en matris där de faktiska värdena placeras längs y-axeln och de predikterade värdena längs x-axeln. Antalet observationer som tillhör varje klass och hur de var klassificerade av modellen fylls sedan i matrisen. De korrekt klassificerade observationerna återfinns på diagonalen, medan felaktigt klassificerade observationer finns på övriga platser i matrisen. Det kan användas bara som det är för att utvärdera modellens prestation men man kan också räkan viktiga mätvärden som F1-score, presicion och recall.

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

K-means är en unsupervised-algoritm som används för klustring. Där vi bestämmer antal klusters(K) som modellen ska dela data i. Algoritmen utgår på att tilldela datapunkter till olika kluster på så sätt så att maximera distansen mellan klusters och minimera den inom klusters.

Tillämpningsområden: Segmentering av patienter för att identifiera riskgrupper, segmentering av kunder för att använda i marknadsföring, eller segmentering av elever för att anpassa undervisningen.

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

De här är tekniker för att behandla kategoriska variabler.

Ordinal encoding handlar om att ersätta kategorier med siffror (från 0) baserad på hierarki eller rankning av kategorier.

One-hot encoding är en teknik då man skapar lika mycket nya variabel som det finns kategorier och sätter 1 eller 0 ifall en observation tillhör denna kategori.

Dummy variables-eknink är samma som one-hor encoding men man tar bort en av kategorier och modellen förstår att en observatin tillhör denna kategori on den inte tillhör någon annan kategori.

**8. 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, det beror på kontexten, datan och målet. Samma data kan betraktas som ordinal eller nominal beroende på kontexten.

**9. 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 en open source framework för snabb och enkel utveckling av webbapplikationer för dataanalys och maskininlärning. Med Streamlit kan man skapa gränssnitt för att visualisera data, experimentera med modeller och dela resultat.

# Självutvärdering

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

Jag tror att det värsta utmaningen var att få Streamlit att funka när du aldrig har arbetat med det (uppdatera Anaconda, ladda ned Visual Studio, mitt i allt inse att du behöver nyare version av Python osv) och preprocessing av bilder. Jag har samarbetat med Filip när det gäller preprocessing av bilder och kan säga att det värdefullt att kunna bolla om saker, dela kod och peppa varandra.

Jag har bestämt mig att komma igång med GitHub så det var en till utmaning: att förstå varför något inte laddas upp, men jag googlade och frågade chat-gpt och löste det 😊

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

Jag hoppas att detta projekt visar att jag uppnår samtliga mål för kursen och

1. Något du vill lyfta fram till Antonio?

Jättebra kurs för att utmana sig. Rapport-delen var sämst för mig.

# Källförteckning

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (Second Edition)*. Sebastopol, CA: O’Reilly Media, Inc.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

Streamlit (2024) https://streamlit.io/