Forudsigelse af VIP-gæsters fremmøde

En datadrevet prædiktionsmodel

Forfattere:

Udviklet i samarbejde med tre andre studerende

1. interne eksamensprojekt

Dato: 6. januar 2025

Antal tegn: 27.454

Indholdsfortegnelse

1 Resumé

Dette projekt fokuserer på udvikling og implementering af en datadrevet prædiktionsmodel for at forudsige fremmødet af VIP-gæster, der spiser guldmenuen hos Viborg F.F. Formålet er at reducere madspild, optimere ressourcestyringen og udnyttelsesgraden ved hjemmekampe.

Med afsæt i CRISP-DM-modellen blev data fra interne og eksterne kilder samlet og analyseret. Tre modeller blev testet – Baseline-modellen, Ridge Regression og Lasso Regression – hvor Lasso Regression blev valgt som den mest egnede. Dog viste modellen begrænsninger i præcision (RMSE på 133.11 og R^2 på 18.4%), hvilket påpeger behovet for yderligere data og forbedringer.

Forretningsanalysen, baseret på Leavitt's organisationsmodel og Business Model Canvas, fremhævede udfordringer som datasiloer og manglende integration. Alexandramodellen viste, at VFF befinder sig på et lavt niveau af datamodenhed, hvilket kræver investering i bedre datasystemer og samt et løft af medarbejdernes IT og dataforståelse.

Implementeringen af modellen blev struktureret efter Kotters 8-trins model for at sikre forandringsledelse.Trods lav præcision har modellen potentiale til at understøtte en datadrevet tilgang, hvis den suppleres med kvalitative undersøgelser af VIP-gæsternes adfærd og yderligere datainitiativer.

Løsningsforslagene omfatter etablering af et centralt datasystem, der samler data fra eksisterende platforme gennem API-integration. Derudover foreslås kvalitative undersøgelser af VIP-gæsternes adfærd, løbende opdateringer af modellen og bedre samarbejde mellem afdelinger for at styrke beslutningsgrundlaget og optimere ressourcestyringen i VIP-loungen

2 Indledning

Viborg F.F. er en fodboldklub i Viborg med et hold i Superligaen. Klubben opererer under branchekoden 931200 (sportsklubber), og fungerer som en sportsvirksomhed med aktiviteter inden for professionel sport, events, talentudvikling og kommercielle initiativer. Derudover har Viborg F.F. et stærkt samfundsengagement gennem deltagelse i lokale arrangementer for at fremme sport og fællesskab.

Ved hjemmekampe på Energi Viborg Arena uddeler Viborg F.F. et fast antal VIP-billetter gennem deres partnere. Klubben mangler overblik over, hvor mange af disse billetter der reelt bliver indløst til hver kamp. Denne usikkerhed skaber udfordringer i planlægningen, og kan føre til ressourcespild, f.eks. i form af unødvendigt klargjorte kuverter og overflødige indkøb.

I forbindelse med de indledende forberedelser til projektet, har data- og marketingafdelingen i Viborg F.F. deltaget i interviews for at belyse deres udfordringer. Her påpeger billet- og stadionansvarlig Palle Nielsen et ønske om bedre indsigt i fremmøde og billetbrug: "… der, hvor det kunne være spændende at dykke ned, er forskellen på, hvad vi kontraktmæssigt giver adgang til for partnerne, hvor mange billetter de afhenter, og hvor mange der rent faktisk kommer." (Bilag 5) Derudover nævner han: "Man tilmelder sig, for at vi ikke skal gøre klar til samtlige, der har kort. Så tilmelder man sig eventen, så vi ved, hvor mange der er. Det handler om madspild, borddækning og alle sådan nogle ting." (Bilag 5)

Disse udfordringer leder os videre til den overordnede problemformulering, der sigter mod at skabe bedre indsigt og optimering af billetadministrationen hos Viborg F.F.

3 Problemformulering

Hvordan kan Viborg F.F. udvikle og implementere en datadrevet prædiktionsmodel til at forudsige andelen af VIP-gæster, der møder op og spiser guldmenuen? Og hvordan kan denne model anvendes til at optimere ressourcestyring og planlægning af fodboldkampe, samtidig med at der sikres en effektiv forandringsledelse i organisationen?

Prædiktionsmodel For at beskrive hvordan Viborg F.F. kan indsamle og analysere relevante data til udviklingen af en datadrevet prædiktionsmodel, som forudser andelen af VIP-gæster, der spiser guldmenuen, fokuserer den tekniske del af prædiktionsmodellen på dataindsamling, datakvalitet og analytiske metoder.

Systemkrav Analyser hvordan prædiktionsmodellen kan integreres i Viborg F.F.'s eksisterende ressourcestyringssystemer, for at optimere planlægningen og minimere spild af ressourcer i forbindelse med fodboldkampe.

Datamodenhed Vurder, hvordan forandringsledelse kan understøtte implementeringen af prædiktionsmodellen, så medarbejdere og ledelse effektivt kan anvende den til at træffe datadrevne beslutninger i daglige arbejdsprocesser og operationel planlægning.

4 Afgrænsning

Projektet er afgrænset til de sæsoner, hvor Viborg F.F. har spillet i Superligaen fra og med 2013/2014. COVID-19-restriktioner i 2019/2020 sæsonerne betød, at disse sæsoner blev udeladt for at sikre relevant datagrundlag. Det vil gøre det lettere at identificere mønstre i gæsternes fremmøde og forbrugsmønstre. Data indsamles fra offentlige kilder såsom Superstats.dk og DMI.dk samt empirisk materiale fra Viborg FF. Der inddrages ikke data fra andre kommercielle eller private aktører. Prædiktionsmodellen udvikles og testes kun ved brug af softwareprogrammet R, da dette er det primære programmeringssprog, som projektgruppen kender til. Dette valg gør det muligt at udnytte eksisterende kompetencer og sikre en effektiv udviklingsproces. Endelig er projektets omfang afgrænset til optimering af ressourcestyring i forbindelse med guldmenuen og VIP-loungen på kampdage. Andre arrangementer eller organisatoriske områder inden for Viborg FFs drift behandles ikke.

4.1 AI Chatbots

Under udarbejdelsen af projektet, er Chatgpt 4.0 blevet anvendt som værktøj til grammatisk og sproglig korrektur, generering af inspiration og idéer, samt støtte til sproglige forbedringer. Modellen er udelukkende anvendt som hjælpemiddel til tekstbaserede opgaver, og har ikke erstattet selvstændig analyse og kritisk tænkning, og endvidere heller ikke til direkte besvarelse af problemformuleringen.

5 Definitioner

I dette projekt defineres VIP-gæster, som dem der har mulighed for at spise guldmenuen. For enkelhedens skyld forkortes Viborg F.F. til VFF.

6 Videnskabsteori og Metode

Pragmatismen er valgt som videnskabelig tilgang, da den fokuserer på praktiske løsninger og vurderer viden ud fra dens anvendelighed i praksis. Sandhed betragtes som funktionel – noget anses for sandt, hvis det fungerer og løser et problem (Egholm, 2017). Tilgangen understøtter vores formål om at udvikle en prædiktionsmodel, der optimerer VFFs ressourcestyring og planlægning af kapdagen. Ved at spørge "Hvad virker?" fremmer vi løsninger, der skaber konkret værdig for VFF.

6.1 Teorivalg

CRISP-DM-modellen (Cross-Industry Standard Process for Data Mining) anvendes til at strukturere projektets udvikling (Lotte, 2024). Modellen sikrer en systematisk tilgang gennem seks faser.
Klubbens forretningsstruktur og kultur analyseres ved hjælp af Leavitt's organisationsmodel, samt Business Model Canvas (Clark, T., 2012), som giver et overblik over nøgleaktiviteter og ressourcer og værdiskabelse. Alexandramodellen (Kølsen, C., 2017, p.15) anvendes til at vurdere VFFs datamodenhed. Denne vurdering er nødvendig for at sikre en effektiv implementering. Til vurdering af forandringsledelse benyttes Kotters 8-trins model, som hjælper med at analysere og understøtte forandringsprocesser ved implementeringen af prædiktionsmodellen (Thorborg, S., 2013, p.13-25).

6.2 Forskningsdesign

Vi benytter abduktion som forskningsdesign, da det kombinerer praktiske resultater med eksisterende data, for at finde den mest sandsynlige forklaring. Ved at anvende data fra DMI.dk, Superstats.dk og

interne data fra VFF udvikler vi en prædiktionsmodel, der fungerer optimalt i praksis.

6.3 Redegørelse for empirisk grundlag

Dataindsamlingen kombinerer kvalitative og kvantitative data. De kvalitative data er baseret på to semistrukturerede interviews med nøglepersoner i VFF. Den første interviewrunde inkluderede medarbejdere fra data- og marketingafdelingen med fokus på datahåndtering, virksomhedens struktur og gæsternes adfærd. Den anden interviewrunde med praktikanter gav indsigt i daglige arbejdsprocesser. Særlige forhold, hvor respondenterne sad foran 60 mennesker med 13 telefoner og kolleger tæt på, kan have påvirket svarenes validitet. De kvantitative data består af sekundære kilder. Interne data fra et Excel-ark, tilrettet af underviseren, indeholder oplysninger om VIP-menu, kapacitet, billetter og fremmøde. De eksterne data fra Superstats.dk og DMI.dk tilføjer kampstatistik og vejrinformation, som supplerer og validerer de kvalitative resultater.

6.4 Dataindsamling og behandling

Før indsamlingen blev interviewguides udarbejdet, og sekundære datakilder som Excel-ark og eksterne kilder blev fundet. Respondenterne blev informeret om formålet med undersøgelsen. Under indsamlingen blev to interviewrunder gennemført: Den første med medarbejdere fra data- og marketingafdelingen, og derefter med praktikanter i dataafdelingen. Samtidig blev sekundære data fra interne og eksterne kilder indsamlet. Efter indsamlingen blev interviews transskriberet og analyseret gennem en indholdsanalyse, hvor centrale temaer blev identificeret. De kvantitative data blev anvendt til at validere og supplere de kvalitative observationer ved at fremhæve mønstre og tendenser.

6.5 Metode i analyseprocessen

Prædiktionsmodel Vi beskriver hvordan VFF kan indsamle og analysere data til udvikling af prædiktionsmodellen. Sekundære kilder som DMI.dk og Superstats.dk kombineres sammen med interne data

fra VFF. Eksterne data omfatter kamp- og vejrudsigtsdata fra Superstats.dk og DMI.dk, som supplerer de interne data.

Systemkrav Integrationen af prædiktionsmodellen i VFFs ressourcestyringssystemer er central og afgørende, for at optimere planlægningen og minimere ressourcespild. Business Model Canvas inddrages til at analysere, hvordan modellen kan tilpasses og skabe værdi for VFFs eksisterende forretningsmodel. Denne tilgang giver et visuelt overblik over nøgleaktiviteter, ressourcer, samarbejdspartnere og værdiskabelse. Derudover undersøges også mulighederne for teknisk implementering og tilpasning af processer for at sikre modellens optimal funktionalitet i praksis.

Datamodenhed Alexandramodellen anvendes til at analysere VFFs datamodenhed, samt organisatoriske struktur og kultur. Kotters 8-trins model bruges til at vurdere de nødvendige forandringsprocesser for implementeringen af prædiktionsmodellen. Kombinationen af modellerne giver en helhedsorienteret forståelse af organisationens evne til at tilpasse sig ændringer og understøtter en effektiv implementering af prædiktionsmodellen.

6.6 Vurdering af validitet og reliabilitet

Validiteten i projektet kan være påvirket af interviewsituationen, hvor pres fra omgivelserne muligvis har påvirket respondenternes svar. For at styrke validiteten er der anvendt triangulering af kvalitative interviews og kvantitative data. Reliabiliteten sikres gennem systematisk transskribering og behandling af interviews, samt anvendelse af sekundære data fra Excel-arket, Superstats.dk og DMI.dk.

6.7 Refleksion over metodevalg

Vores metodiske tilgang kombinerer pragmatisme, abduktion og en trianguleret dataindsamling for at udvikle en praktisk og robust prædiktionsmodel. Vi anerkender de potentielle udfordringer ved interviewsituationen og afhængigheden af tilgængelige data. I projektet anvender vi vejrdata samt data fra VFF. På grund af begrænsninger i udvalget af variabler, vil prædiktionsmodellen derfor også være begrænset i sin præcision og anvendelighed.

7 Analyse

7.1 Forretningsforståelse

VFF's organisationsstruktur, kultur og forretningsmodel analyseres ved hjælp af Leavitt's organisationsmodel og Business Model Canvas. Modellen opdeler virksomheden i fire nøgleelementer: Struktur, personer, teknologi og opgaver.

7.1.1 VFF-organisation og kultur

VFFs formelle organisationsstruktur, er baseret på funktionsprincippet (Eilersen, Simon B., 2024), hvor opgaver fordeles på specialiserede områder som sport, administration, økonomi, salg, osv. (Kontaktinfo og åbningstider, 2024). Virksomheden har mellem 20-25 medarbejdere ansat i administrationen, og operationelle afdelinger, samt lidt over 200 frivillige, som hjælper på kampdage (Bilag 5).

Direktionen træffer strategiske beslutninger, mens afdelingerne håndterer daglige opgaver, og tilpasser sig konkrete udfordringer. IT-chef Daniel Behr forklarer: "Det er lidt forskelligt fra afdeling til afdeling…" (Se Bilag 6).

Den uformelle struktur i virksomheden bygger på fælles værdier og relationer, som styrker samarbejdet og skaber sammenhæng. Symbolismen i den grønne farve og klubbens logo, skaber loyalitet blandt både medarbejdere og fans. Missionen "Vi vil skabe fællesskaber og oplevelser, der er blandt de bedste i Danmark" (Bilag 5) og visionen "Forene mennesker for at realisere det grønne potentiale" (Bilag 5) viser virksomhedens værdigrundlag.

Medarbejderne en afgørende rolle for VFFs succes. IT-chef Daniel Behr udtaler: "Vi ansætter folk, som er specialister inden for noget særligt." (Bilag 6), hvilket viser, at VFF tilbyder initiativer som IT-uddannelsesdage, hvor formålet er at hæve medarbejderkompetencerne.

Teknologi er vigtigt for VFFs drift. Virksomheden benytter systemer som Joomla til hjemmesiden Vff.dk, Shopify til Kløvershoppen, Eventii til billetsalg, CRM-platforme til partnerhåndtering, Playable

til konkurrencer, MailChimp til mail (Tea Nørgaard, Marketingpræsentation 2024). IT-chef Daniel Behr fremhæver: "Der er ekstremt mange forskellige systemer, hvor der er noget data i." (Bilag 6).

7.1.2 VFFs Business Model Canvas

I dette afsnit præsenteres en konklusion af VFFs Business Model Canvas. For en detaljeret gennemgang, samt figur henvises der til bilag 1.

Analysen viser, at virksomheden har en forretningsmodel, der effektivt understøtter klubbens nøgleaktiviteter, og imødekommer interessenternes forskellige behov. Klubben henvender sig til forskellige kundesegmenter, herunder sponsorer, partnere, fans, studerende og familier, hvor hvert segment spiller en rolle i at sikre virksomhedens økonomiske succes (Bilag 1). De primære værdifaktorer er kampe, oplevelser, netværksmuligheder, som understøttes af både digitale og fysiske kanaler (Bilag 1).

På baggrund af projektets problemstilling og problemformulering, vurderes det, at en central udfordring i klubbens nuværende forretningsmodel, er manglen på præcise data, herunder omkring VIP-gæster til fremmøde, samt deres adfærd. Uden et klart billede over, hvor mange VIP-gæster der møder op og benytter sig af tilbud som guldmenuen, risikerer klubben spild af både tid og ressourcer, hvilket kan påvirke den økonomiske situation negativt.

7.2 Datamodenhed

Klubben har gjort fremskridt det seneste år, hvilket praktikant Mohammed fremhæver: "Altså jeg synes, at datamodenhedsmæssigt … føler jeg, at Viborg FF har rykket sig en del fra sidste år her i forhold til datamæssigt" (Bilag 6). Trods af fremskridtene, står klubben stadig overfor udfordringer, som datasiloer og manglende integration mellem afdelinger. IT-chef Daniel Behr forklarer: "… vi har ekstremt mange forskellige systemer hvor der er noget data i, og det der med at få det integreret lidt sammen så det ikke bliver sådan meget silooplevelse det er en udfordring." (Bilag 6). Alexandramodellen anvendes til at analysere virksomhedens datamodenhed og identificere udviklingspotentiale gennem primære datakilder. (Bilag 5 og 6)

7.2.1 Alexandra modellen

Den fulde analyse af VFFs datamodenhed ved hjælp af Alexandramodellen findes i bilag 2. Analysen viser, at virksomheden befinder sig i fase to, "Lære om forretningen," hvor data bruges til at forstå interne processer og kundebehov (Kølsen, C. et al, 2017). Marketingansvarlig Daniel Behr forklarer: "… vi vil gerne få mange flere beslutninger taget på baggrund af datagrundlaget og ikke så meget af mavefornemmelser og oplevelser" (Bilag 5). Samtidig oplever VFF udfordringer som datasiloer og varierende brug af data mellem afdelingerne. Behr bemærker: "Det er lidt forskelligt fra afdeling til afdeling" (Bilag 6).

Analysen konkluderer, at VFF har mulighed for øge deres datamodenhed, og dermed styrke grundlaget for bedre forretningsbeslutninger, anbefales det at etablere et data warehouse, nedbryde datasiloer, udvikle prædiktionsmodeller og automatisere processer. Disse tiltag vil gøre VFF mere datadrevet, øge datamodenheden og forbedre virksomhedens forretningsbeslutninger.

7.3 Prædiktiosmodellen

7.3.1 Dataanalyse

Projektet fokuserer på at forudsige efterspørgslen på VIP-guldmenuer, hvilket blev omsat til en konkret datamining-opgave. Datasættet blev oprettet ved at kombinere informationer fra Excel-ark, kampdata fra Superstats.dk og vejrobservationer fra DMI's API, med kampdata som fælles reference for at sikre sammenhæng i analysen.

7.3.2 Dataforståelse og forberedelse

Den afhængige variabel, "guld_menu_stk", repræsenterer antallet af afhentede guldmenuer, og var analysens fokus. Forklarende variabler inkluderede vejrforhold, tilskuertal, kampdato og modstanderes styrke, udvidet med feature engineering som point fra de sidste tre kampe og ugedag.

Datasættet blev renset og transformeret i R, hvor manglende værdier blev håndteret med medianimputering for numeriske variabler og "Unknown" for kategoriske. Korrelationsanalyser identificerede lineære

sammenhænge, hvilket reducerede risikoen for multikollinearitet. Outliers blev ikke vurderet, hvilket kan påvirke modellens præcision, men dette prioriteres i fremtidige justeringer

7.3.3 Explorative analyse

Den eksplorative analyse inkluderede histogrammer, scatterplots og korrelationsmatrixer for at identificere mønstre og sammenhænge. Stærkt korrelerede variabler blev fjernet for at reducere multikollinearitet. (Se koden)

7.3.4 Valg af metode

En struktureret tilgang blev anvendt til at evaluere modellens ydeevne. Datasættet blev opdelt i trænings- og testdata (2/3 træning og 1/3 test) for at sikre, at modellen kunne generalisere til nye data. Desuden blev 10-fold krydsvalidering brugt til at optimere parametrene for Ridge og Lasso Regression og reducere risikoen for overfitting.

7.3.5 Konkurrende modeller

I projektet blev tre modeller testet, for at finde den mest effektive metode til at forudsige efterspørgslen. Den første model, Baseline-modellen, forudsagde middelværdien uden forklarende variabler (RMSE = 147.49). Den anden model, Ridge Regression, inkluderede alle forklarende variabler og reducerede RMSE til 125.67, hvilket forbedrede præcision sammenlignet med baseline-modellen. Den tredje model, Lasso Regression, opnåede en RMSE på 133.11, og forenklede modellen ved at udvælge de mest relevante variabler.

7.3.6 Modeludvælgelse

Den endelige model blev valgt ud fra en balance mellem præcision og simplicitet. Lasso Regression blev valgt på grund af dens evne til at reducere variabler uden væsentligt præcisionstab. Modellen reduc-

erede RMSE med ca. 10% sammenlignet med baseline, og inkluderede seks nøglevariabler, hvor de mest betydningsfulde var `guldmenu_sidste_møde`, `antal_tilskuere` og `luftfugtighed`.

7.3.7 Diskussion

Modellen opnåede en RMSE på 133.11 og en forklaringsgrad (R^2) på 18.4%, hvilket viser, at væsentlige forklarende faktorer mangler, og modellen bør forbedres med yderligere data. Modellen giver dog indsigt i faktorer som ugedage og modstanderens historik, der påvirker efterspørgslen. Fx viser data, at kampe mod bundklubben Vejle, skaber høj efterspørgsel på guldmenuer, hvilket kan skyldes sociale relationer eller arrangementer. Denne konklusion er dog statistisk usikker, da VFF kun har mødt Vejle fire gange i Superligaen mellem 2013 og 2023.

For at gøre modellen mere anvendelig anbefales det at:

- Indsamle data om netværksaktiviteter og events.
- Analysere VIP-gæsters adfærd adfærdsmønstre.
- Løbende opdatere modellen med nye data.
- Udvikle en ny variabel for sponsorers brugeradfærd i forhold til guldmenuer.

7.3.8 Konklusion

Lasso Regression-modellen viser potentiale til at forudsige efterspørgslen på VIP-guldmenuer, men kræver yderligere variabler og løbende opdateringer for at forbedre præcisionen, og gøre den praktisk anvendelig.

Konklusionen er, at modellen endnu ikke præcis nok til praktisk brug, og kræver yderligere data. Indtil da bør medarbejderne fortsat estimere antallet af afhentede guldmenuer.

7.4 Forandringsledelse

Implementeringen af en prædiktionsmodel i VFF til at forudsige VIP-gæsters fremmøde, kræver, at udfordringerne som manglende data og relevante variabler om gæsternes adfærd påtales. Trods udfordringerne kan en Bottom-up tilgang kombineret med en adaption-strategi støtte en succesfuld forandringsledelse.

Virksomhedens kultur, der balancerer mellem funktionalisme og symbolisme, spiller en vigtig rolle i implementeringen. Funktionalismen sikrer klare strukturer og roller, mens symbolismen kan fremme medarbejderengagementet.

Ved hjælp af Kotters 8-trins model kan VFF skabe en ramme for implementeringen af prædiktionsmodellen og tackle udfordringer med manglende data. Følgende afsnit opsummerer Kotters 8-trins model, mens en mere detaljeret analyse af hvert trin findes i bilag 3.

7.4.1 Kotters 8-trins model

Kotters 8-trins model strukturerer implementeringen af en prædiktionsmodel i VFF. Processen starter med en "Burning platform", hvor Palles Nielsens bemærkninger om madspild og unødvendigt arbejde i VIP-loungen, fremhæver behovet for forandring. Dette motiverer medarbejderne og skaber en fælles forståelse for nødvendigheden af modellen.

En styrende koalition sammensættes af nøglepersoner som Daniel Behr, Tea N. Pedersen, Palle Nielsen og Christian Storvik. Daniel Behr bidrager med teknisk indsigt og dataudvikling, Tea N. Pedersen assisterer med dataindsamling, Palle Nielsen håndterer billetter og daglige operationer, og Christian Storvik repræsenterer partnernes interesser.

Visionen, "Sammen skaber vi en stærkere VFF, hvor prædiktionsmodeller leverer præcise dataindsigter, reducerer spild og optimerer gæsternes oplevelse," (Bilag 5), kommunikeres gennem møder og nyhedsbreve for at engagere medarbejderne og fremme forandringen.

7.5 CRISP-DM

Projektet blev struktureret efter CRISP-DM-modellen, for at sikre en fokuseret tilgang, hvor de seks faser guidede dataanalysen, og gjorde den relevant for problemstillingen.

Business understanding Virksomhedens udfordringer og behov blev identificeret gennem primære datakilder. Selvom VIP-gæster ikke blev betragtet som en problematik af VFF, fremhævede data og underviserens vejledning dette som et fokusområde for projektet.

Data understanding Vi samlede data som VIP-guldmenu fra Excel-ark, kampdata fra Superstats.dk via webscraping og meteorologiske observationer fra DMI. Disse blev kombineret baseret på kampdatoer for at sikre sammenhæng og relevans. Den valgte outputvariabel var antallet af VIP-guldmenuer, og forklarende variabler inkluderede vejrdata, kampstatistikker og faktorer som fx ugedag.

Data preperation Datasættet blev gennemgået og renset ved fjerne irrelevante data, udfylde manglende værdierog outliers. Nye variabler som tilskuertal ved sidste møde, point fra de sidste tre kampe og årstid blev tilføjet. Datasættet var derefter klar til modellering.

Modelling
Tre modeller blev evalueret: baseline-modellen, Ridge Regression og Lasso Regression. Lasso Regression blev valgt, da den reducerede antallet af variabler uden væsentligt præcisionstab, men med en begrænset RMSE på 133.11 og en forklaringsgrad på kun 18.4%, hvilket indikerer manglende faktorer mangler i datasættet.

Evaluation Resultaterne viste, at modellen ikke kunne levere præcise forudsigelser af VIP-gæsters fremmøde, og guldmenuer. Begrænsninger i datakvalitet, samt mængde af data reducerede modellens præcision betydeligt.

Deployment Implementeringsfasen fokuserede på integrering af prædiktionsmodellen i VFFs arbejdsgange for at skabe en datadrevet tilgang. Med afsæt i Kotters 8-trins model blev der etableret en fælles forståelse for modellens potentiale (Trin 1) og en styrende koalition på tværs af nøgleafdelinger blev oprettet (Trin 2). Et centralt datasystem blev udviklet for at samle data og nedbryde siloer, mens medarbejdere blev trænet i brugen af modellen (Trin 5). Modellens lave præcision understreger behovet for yderligere

data og en trinvis implementering for at styrke datakvalitet, samarbejde og en datadrevet kultur (Trin 6-8)

8 Anbefalinger til VFF

8.1 Prædiktionsmodel og Machine learning

For at øge modellens anvendelighed anbefales det, at VFF indsamler data om netværksaktiviteter og events samt analyserer VIP-gæsternes adfærd, for at identificere årsager til fremmøde. Dataafdelingen bør desuden løbende opdatere modellen med nye data for at forbedre dens præcision. Der anbefales, at VFF laver en prædiktion af hvor mange der skal bespises til en kommende hjemmebanekamp hver dag i 14 dage op til kampen, så der kommer en løbende opdatering over dette, og dermed forberede indkøbere og eventplanlæggere til eventet. VFF bør supplere prædiktionerne med Palle Nielsens' erfaringer og input således at VFF tager beslutningerne, som ikke udelukkende er baseret på data, men er et sammenspil mellem data og allerede kendte erfaringer og informationer. VFF bør styrke dette samarbejde.

8.2 Kvalitativ undersøgelse

VFF anbefales at lave en kvalitativ undersøgelse, og undersøge adfærden hos VIP-gæsterne i forhold til hvilke faktorer spiller ind, når de skal beslutte om de vil komme til eventet eller ej.

8.3 Systemkrav

For at implementere en effektiv prædiktionsmodel bør VFF. fokusere på at centralisere og integrere deres eksisterende systemer. Klubben anvender allerede Eventii til billetsalg, CRM-systemer til partnerhåndtering og MailChimp til e-mailkommunikation, men mangler en sammenhængende datainfrastruktur.

Det anbefales at etablere et centralt data warehouse som Microsoft Azure, der kan samle data fra de eksisterende systemer. Samtidig bør API'er anvendes til at forbinde systemerne og sikre, at data automatisk

opdateres. Dette vil minimere manuel datahåndtering, reducere fejl og sikre, at modellen har adgang til opdaterede og relevante data, hvilket også efterspørges af IT-Chef Daniel Behr (Bilag 6).

Til analyser og visualisering anbefales Microsoft Power BI, som gør det muligt at præsentere modellens resultater i et visuelt og brugervenligt format. Dette vil understøtte bedre beslutningstagning og fremme en datadrevet tilgang i VFF.

Systemopsætningen bør være fleksibel, så nye datakilder som sociale faktorer eller netværksaktiviteter kan integreres i fremtiden. Ved at implementere disse løsninger kan VFF reducere spild, forbedre ressourcestyringen og styrke deres datadrevne arbejdsgange.

8.4 Datamodenhed

For at opnå en bedre datamodenhed anbefales det, at virksomheden indfører mere uddannelse af deres medarbejdere, så VFF øger deres datakultur og databevidsthed. Det anbefales også, at VFF undersøger muligheden for en centraliseret platform, hvor medarbejderne kan dele data med hinanden. Virksomheden bør også etablere af et data warehouse, arbejde på at nedbryde datasiloer på tværs af afdelinger, udvikle prædiktionsmodeller og automatisere processer.

9 Konklusion

Problemformuleringen i projektet stiller spørgsmålet: ”Hvordan kan Viborg F.F. udvikle og implementere en datadrevet prædiktionsmodel til at forudsige andelen af VIP-gæster, der møder op og spiser guldmenuen? Hvordan kan denne model optimeres til at understøtte virksomhedens ressourcestyring og reducere spild ved fodboldkampe?”

Ved at kombinere analyser af forretningsforståelse, datamodenhed og implementering gennem forandringsledelse, er projektets problemstilling blevet behandlet ud fra flere perspektiver. Hermed præsenteres en samlet konklusion:

Forretningsanalysen viste, at VIP-segmentet er centralt for at reducere madspild og forbedre kundetilfredshed. VFFs interne strukturer og kultur understøtter delvist en datadrevet tilgang, men der er behov for yderligere tilpasning.

Vurderingen af datamodenhed placerede Viborg F.F. på et niveau 2. Klubben bruger data til beslutningstagning, men udfordres af datasiloer og manglende integration. Etablering af et data warehouse og avancerede analyser anbefales. VFF bør fortsat vejlede/uddanne medarbejderne i IT.

Tre prædiktionsmodeller blev testet. Lasso Regression blev valgt på grund af dens balance mellem enkelthed og præcision, men modellens RMSE på 133.11 og forklaringsgrad på 18.4% viser, at yderligere data og variabler er nødvendige for at forbedre præcisionen.

Implementeringen blev struktureret med Kotters 8-trins model. Fokus var på at skabe en "burning platform", etablere en styrende koalition og træne medarbejdere. Visionen om en datadrevet tilgang blev kommunikeret bredt for at sikre engagement og accept.

Modellen viser potentiale, men dens succes afhænger af bedre datagrundlag, organisatorisk tilpasning og en langsigtet strategi. VFF kan med en trinvis implementering styrke deres datadrevne kultur og forbedre ressourcestyringen. Projektets løsningsforslag er derfor, at VFF bør gennemføre en kvalitativ undersøgelse af guldmenu-gæsterne for at afdække de adfærdsmønstre, der påvirker deres fremmøde. Dette vil give VFF en bedre forståelse af årsagerne bag gæsternes deltagelse, og muliggøre løsninger, der kan øge udnyttelsesgraden, herunder finde nye variabler.

10 Literaturliste

AI

OpenAI. (2024). ChatGPT (4.0). https://chatgpt.com/

Bøger

Clark, T. et al. (2012). Business model generation: En håndbog for nytænkere, banebrydere og rebeller. Gyldendal Business. (Originalværk udgivet 2012)

Egholm, L. (2017). Videnskabsteori: Perspektiver på organisationer og samfund. Hans Reitzel. (Originalværk udgivet 2014)

Thorborg, S. (2013). Forandringsledelse: En grundbog. Hans Reitzel. (Originalværk udgivet 2013)

WWW-dokumenter

Business Model Canvas skabelon. (2024). Business Model Canvas skabelon. https://modernbusiness.dk/business-model-canvas-skabelon/#google_vignette Ejer af skabelon modernbusiness.dk

CVR API. (2024). Sportsklubber. Lokaliseret den 2024 på https://cvrapi.dk/branche/dk/931200

Kontaktinfo og åbningstider. (2024). Kontaktinfo og åbningstider. Lokaliseret den 19. december 2024 på https://www.vff.dk/viborg-f-f/kontaktinfo

Kølsen, L. et al. (2017). Find vej i din dataindsats. Alexandra instituttet for Industriens Fond. https://alexandra.dk/wp-content/uploads/2020/09/Alexandra-Instituttet-BDBA-Find-vej-i-din-dataindsats.pdf. Lokaliseret d. 17/12 2024.

Kølsen, C. et al (2017). Find vej i din dataindsats. Alexandra Instituttet for Industriens Fond. https://eadania.mrooms.net/pluginfile.php/636535/mod_page/intro/alexandra%20institut.pdf

Viborg fodbold forening. (2024.). Viborg fodbold forening. Lokaliseret den 19. december 2024 på https://www.vff.dk

Undervisningsmaterialer B. Eilersen, Simon. (2024, 13. september). Data i en forretningskontekst: Organisation og forretningsmodeller . Erhvervsakademi Dania. Soelberg Lotte. (2024, 4. september). Rigtig mødetid: Fællessamling.

Præsentationer fra VFF Billetsalg, Dataafdelingen, Marketing, Frederik, Mohammed og Olga.

11 Bilagsoversigt

- Bilag 1: Business model Canvas
- Bilag 2: Alexandra modellen
- Bilag 3: Kotters 8 trins model
- Bilag 4: CRISP-DM
- Bilag 5: Transskribering af interview 1
- Bilag 6: Transskribering af interview 2
- Bilag 7: Skematisering i Excel af interview 1
- Bilag 8: Skematisering i Excel af interview 2
- Bilag 9: Dataanalyse af interview 1
- Bilag 10: Dataanalyse af interview 1
- Bilag 11: Koden - Findes i Gruppe_vff.qmd se nedenfor

```
#==============================================================================
#                                  Introduktion
#==============================================================================
# Denne kode udfører en omfattende dataanalyse og databehandling med henblik på
# at opbygge et datasæt, der kan anvendes til at modellere og forudsige
# VIP-guldmenuafhentninger for Viborg FF (VFF). Workflowet omfatter alt fra
# dataindsamling og rengøring til oprettelse af nye variabler, udførelse af
# eksplorativ dataanalyse (EDA) og forberedelse til modellering. Vi benytter
# flere datakilder, herunder Excel-filer, webscraping fra Superstats og vejrdata
# fra DMI.

# Workflow

# 1. Dataindsamling og første rengøring
# Data fra forskellige kilder indlæses, og vi sikrer ensartethed i kolonnenavne
# og datatyper. Manglende værdier håndteres med strategier som median og glidende
# gennemsnit.

# 2. Eksplorativ dataanalyse (EDA)
# Visualiseringer som boxplots, histogrammer og korrelationsmatricer bruges til
# at identificere mønstre, outliers og sammenhænge mellem variabler. VIF-analyse
# bruges til at fjerne multikollinearitet.

# 3. Oprettelse af nye variabler
# Vi opretter variabler baseret på tidligere møder, klassificering af modstandere
# og årstider for at styrke modellens forklaringskraft.

# 4. Databehandling og forberedelse til modellering
```

```
# Datasættet trimmes, irrelevant data fjernes, og variabler skaleres eller kodes,
# så de er klar til at blive anvendt i forskellige modeller.

# 5. Modellering
# Vi implementerer og evaluerer modeller som baseline, Ridge og Lasso regression
# samt vurderer præstationen ved hjælp af metrikker som RMSE, MAE og R². Derudover
# analyserer vi residualplots for at sikre modellernes robusthed.

# 6. Prediction
# Vi anvender de trænerede modeller til at lave forudsigelser for kommende kampe,
# baseret på input som vejrdata og tidligere kampresultater.

# Notation og kodestandard:
# - Vi bruger lower_case til alle variabel- og kolonnenavne.
# - Navne er korte, men beskrivende, og specialtegn undgås for at sikre konsistens
#   og forståelighed.
# - Kommentering fokuserer på at forklare rationalet bag beslutninger og gør det
#   lettere for læseren at følge workflowet.

#==============================================================================
#                                   Pacman
#==============================================================================

# Vi bruger pacman-pakken til at administrere og loade nødvendige R-pakker.
# Fordele ved pacman:
# 1. Automatisk installation: Hvis en pakke mangler, installerer pacman den for os.
# 2. Effektivitet: Alle pakker loades med én funktion, så vi slipper for individuelle 'lib
# 3. Struktur: Giver en samlet oversigt over alle nødvendige pakker for projektet, hvilket
```

```r
# Pacman sikrer, at vores miljø har de nødvendige værktøjer til datahåndtering, visualiser
# Her loader vi pakker, der er nødvendige for de forskellige dele af workflowet.

pacman::p_load(
  httr,          # Til at hente data fra API'er som DMI. API'er giver ofte data, som ikke l
  jsonlite,      # Gør det muligt at konvertere JSON-data (et typisk API-format) til tabel
  tidyverse,     # En samling af pakker (fx ggplot2 og dplyr) til datamanipulation og visu
  rvest,         # Til web scraping, fx for at hente data fra Superstats. Vi bruger dette
  readxl,        # Til indlæsning af Excel-filer, da mange organisationer gemmer data i Ex
  tidyr,         # Til datatransformation, fx bredt til langt format eller omvendt. Gør ko
  glmnet,        # Implementerer Ridge og Lasso regression, som er avancerede metoder til
  boot,          # Bruges til krydsvalidering, som sikrer robust evaluering af vores model
  dplyr,         # Bruges til datafiltrering, transformation og aggregering. En del af tid
  stringr,       # Til avanceret tekstbehandling, fx når vi renser eller omstrukturerer te
  lubridate,     # Gør dato- og tidsmanipulation nemt, fx konvertering mellem datoformater
  ggplot2,       # Visualisering af data. Hjælper med at skabe professionelle grafer og di
  car,           # Giver avancerede statistiske værktøjer som VIF, der bruges til at opdag
  corrplot,      # Bruges til at visualisere korrelationsmatricer. Gør det let at identifi
  slider         # Til beregning af glidende gennemsnit og vinduer, fx når vi analyserer t
)

# Hvorfor gøre dette her i starten af koden?
# 1. Det giver et samlet overblik over alle afhængigheder, hvilket gør projektet lettere a
# 2. Hvis nogen skal køre koden på deres egen maskine, vil denne del sikre, at alle nødven
# 3. Hvis nye pakker bliver nødvendige under udviklingen, kan de nemt tilføjes til denne s

# Bemærk: Før pacman kan bruges, skal det installeres manuelt én gang med:
# install.packages("pacman")
```

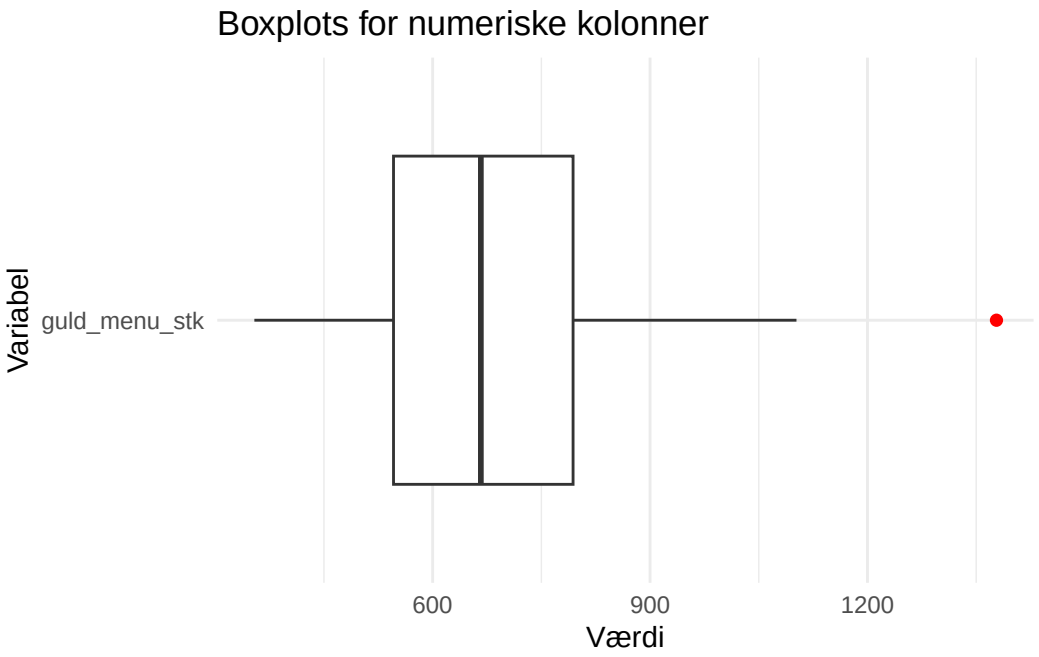
```r
#==============================================================================
#                        1. Dataindsamling og første rengøring
#==============================================================================

# ------------------------------------------------------------------------------
# Læs og rengør excel_guld
# ------------------------------------------------------------------------------

# Vi indlæser en Excel-fil, der indeholder oplysninger om VIP-guldmenuer.
# Datasættet kræver oprydning for at gøre det egnet til analyse.
# VIGTIGT: Excel-filen skal lægge i en datamappe, der er placeret samme sted som r-filen

excel_guld <- read_excel("data/Guld.xlsx") |>
  dplyr::select(-Gule_poletter_stk, -Kamp) |>  # Fjern irrelevante kolonner.
  na.omit()  # Fjern rækker med manglende værdier for at undgå fejl i analysen.

# Konvertering af datoformat:
# Datoen i datasættet er i tekstformat ("dd.mm.yyyy"). Vi konverterer dette til
# et standardiseret R-datoformat, der kan bruges til beregninger og visualisering.

excel_guld <- excel_guld |>
  mutate(dato = as.Date(Dato, format = "%d.%m.%Y"))

# Hvis nogle datoer stadig ikke er læsbare (fx på grund af Excel-serienumre),
# konverteres disse til gyldige datoer med 'origin = "1899-12-30"'.

excel_guld$dato[is.na(excel_guld$dato)] <- as.Date(as.numeric(excel_guld$Dato[is.na(excel_g
                                                    origin = "1899-12-30")
```

```r
# Oprydning af datasættet:
# Vi fjerner den originale Dato-kolonne og reorganiserer variablerne for at gøre
# datasættet mere overskueligt.

excel_guld <- excel_guld |>
  dplyr::select(-Dato) |>  # Fjern den gamle Dato-kolonne.
  relocate(dato, .before = Guld_menu_stk)  # Flyt 'dato'-kolonnen til starten.

# Konvertering af variabler:
# 'Guld_menu_stk' konverteres til numerisk format og afrundes til nærmeste heltal
# for at sikre konsistens og forberede til analyse.

excel_guld <- excel_guld |>
  dplyr::select(-Antal_bestilte, -Antal_max) |>  # Fjern yderligere irrelevante kolonner.
  mutate(guld_menu_stk = as.integer(as.numeric(Guld_menu_stk))) |>
  dplyr::select(-Guld_menu_stk)  # Fjern originalkolonnen efter konvertering.

# Vis et hurtigt overblik over datasættets struktur og variabler:
# Funktionen 'glimpse' giver os en idé om datastrukturen og datatyperne.

glimpse(excel_guld)
```

```
Rows: 98
Columns: 2
$ dato          <date> 2013-07-20, 2013-08-09, 2013-08-25, 2013-09-01, 2013-09~
$ guld_menu_stk <int> 470, 791, 644, 768, 682, 748, 1378, 514, 412, 535, 680, ~
```

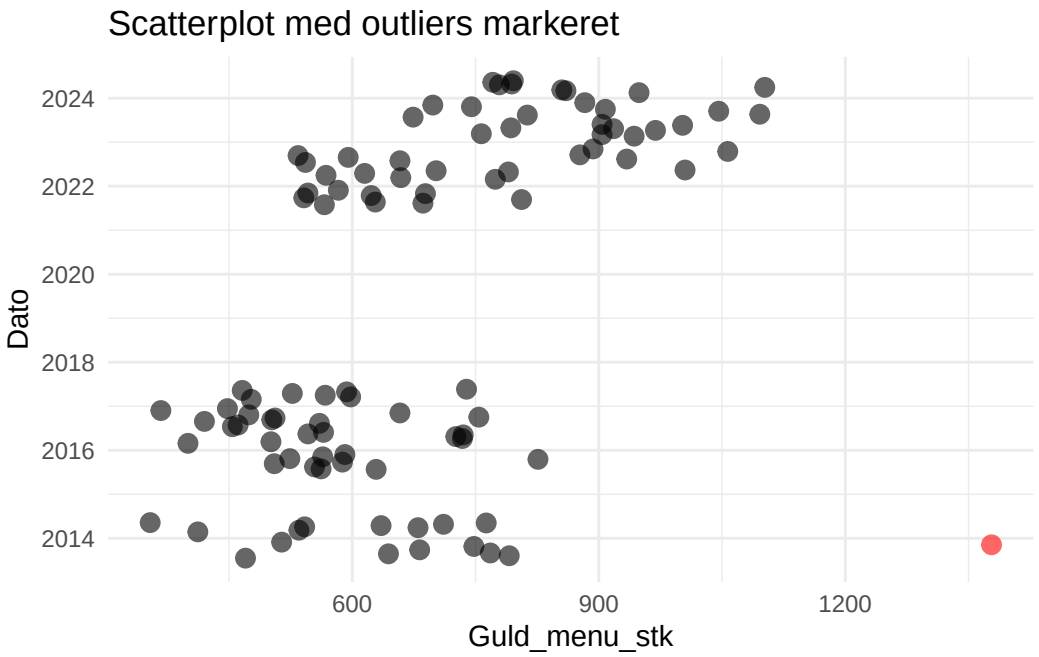
```r
# ------------------------------------------------------------------------------
# Visualisering af outliers
# ------------------------------------------------------------------------------

# Identifikation af outliers:
# Vi bruger interkvartilområdet (IQR) til at finde observationer, der ligger
# uden for 1.5 * IQR fra kvartilerne. Disse observationer betragtes som outliers.

find_outliers <- function(x) {
  iqr <- IQR(x, na.rm = TRUE)  # Beregn IQR.
  lower_bound <- quantile(x, 0.25, na.rm = TRUE) - 1.5 * iqr  # Nedre grænse.
  upper_bound <- quantile(x, 0.75, na.rm = TRUE) + 1.5 * iqr  # Øvre grænse.
  x < lower_bound | x > upper_bound  # Returner TRUE for outliers.
}

# Vi identificerer outliers i 'Guld_menu_stk' og viser dem som en tabel for
# bedre forståelse.

outliers <- excel_guld |>
  filter(find_outliers(guld_menu_stk)) |>  # Filtrer rækker med outliers.
  dplyr::select(dato, guld_menu_stk)  # Vælg relevante kolonner.

# Visualisering af outliers:
# Vi bruger boxplots til at vise fordelingen af numeriske variabler og markerer
# outliers med røde prikker.

excel_guld |>
  pivot_longer(cols = where(is.numeric), names_to = "variable", values_to = "value") |>
```

```
  ggplot(aes(x = variable, y = value)) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 2) +
  coord_flip() +  # Roter diagrammet for bedre læsbarhed.
  labs(
    title = "Boxplots for numeriske kolonner",
    x = "Variabel",
    y = "Værdi"
  ) +
  theme_minimal()
```




```
# Scatterplot:
# Vi plotter 'Guld_menu_stk' mod 'dato' og markerer outliers med rød farve.


excel_guld |>
  mutate(is_outlier = find_outliers(guld_menu_stk)) |>
  ggplot(aes(x = guld_menu_stk, y = dato)) +
```

```
  geom_point(aes(colour = is_outlier), alpha = 0.6, size = 3) +
  scale_colour_manual(values = c("FALSE" = "black", "TRUE" = "red"), guide = "none") +
  labs(
    title = "Scatterplot med outliers markeret",
    x = "Guld_menu_stk",
    y = "Dato"
  ) +
  theme_minimal()
```




```
# Udskrivning af identificerede outliers.
print(outliers)
```

```
# A tibble: 1 x 2
  dato       guld_menu_stk
  <date>             <int>
1 2013-11-08          1378
```

```r
# Behandling af outliers:
# Vi markerer outliers som NA, så de kan håndteres senere.

excel_guld <- excel_guld |>
  mutate(
    guld_menu_stk = ifelse(
      find_outliers(guld_menu_stk),  # Hvis observationen er en outlier,
      NA,                            # marker den som NA.
      guld_menu_stk                  # Behold den ellers som den er.
    )
  )

# Håndtering af manglende værdier:
# Vi erstatter NA med gennemsnittet af de øvrige værdier i 'Guld_menu_stk'.

excel_guld <- excel_guld |>
  mutate(
    guld_menu_stk = ifelse(
      is.na(guld_menu_stk),
      round(mean(guld_menu_stk, na.rm = TRUE)),  # Erstat med afrundet gennemsnit.
      guld_menu_stk
    )
  )

# Endelig tjek af datasættet:
# Vi udskriver det opdaterede datasæt for at sikre, at ændringerne er korrekte.

print(excel_guld)
```

```
# A tibble: 98 x 2
   dato       guld_menu_stk
   <date>             <dbl>
 1 2013-07-20           470
 2 2013-08-09           791
 3 2013-08-25           644
 4 2013-09-01           768
 5 2013-09-27           682
 6 2013-10-25           748
 7 2013-11-08           682
 8 2013-11-30           514
 9 2014-02-23           412
10 2014-03-09           535
# i 88 more rows
```

```r
# ================================================================================
#                       Læs og rengør Superstats-data
# ================================================================================


# --------------------------------------------------------------------------------
# Funktion til at validere en tabel
# --------------------------------------------------------------------------------
# Denne funktion tjekker, om en tabel indeholder gyldige data, som vi kan bruge.
# Kriterierne er:
# 1. Tabellen skal have rækker (dvs. ikke være tom).
# 2. Tabellen skal indeholde kolonnerne 'X3', 'X4' og 'X5'.
# 3. Mindst én af kolonnerne 'X3', 'X4' eller 'X5' skal indeholde gyldige værdier.
valider_tabel <- function(tabel) {
  nrow(tabel) > 0 &&  # Tjekker om tabellen ikke er tom
```

```r
    all(c("X3", "X4", "X5") %in% names(tabel)) &&  # Tjekker om kolonnerne findes
    any(tabel$X3 != "" | tabel$X4 != "" | tabel$X5 != "")  # Tjekker om der er data i mind
}

# ------------------------------------------------------------------------------
# Funktion til at hente og filtrere tabeller for en enkelt sæson
# ------------------------------------------------------------------------------
# Denne funktion henter data fra Superstats.dk for en specifik sæson.
# Den filtrerer tabellerne og tilføjer sæsonoplysninger til dataene.
# Input: Årstal (fx 2013 for sæsonen 2013/2014).
# Output: En tibble med data for den pågældende sæson.
hent_superstats_data <- function(år) {
  message("Har hentet data fra sæsonen: ", år, "/", år + 1)  # Statusbesked

  # Byg URL'en for den specifikke sæson
  url <- paste0("https://superstats.dk/program?aar=", år, "%2F", år + 1)

  # Hent HTML-indhold og udtræk tabeller
  alle_tabeller <- read_html(url, encoding = "UTF-8") |>
    html_elements("table") |>  # Finder tabellerne i HTML
    html_table(header = FALSE, convert = FALSE)  # Konverterer tabellerne til R-objekter

  # Filtrer tabellerne, så kun de gyldige bliver tilbage
  gyldige_tabeller <- purrr::keep(alle_tabeller, valider_tabel)  # Beholder kun tabeller,

  # Tilføj sæsonoplysninger og kombiner tabellerne til én tibble
  purrr::map_dfr(gyldige_tabeller, ~ as_tibble(.x) |> mutate(sæson = paste0(år, "/", år +
}
```

```r
# ------------------------------------------------------------------------------
# Hent og rengør data for alle sæsoner (2013-2023)
# ------------------------------------------------------------------------------
# Vi henter data fra Superstats.dk for sæsonerne 2013-2023 og udfører følgende trin:
# 1. Fjern irrelevante kolonner ('X1', 'X6', 'X7', 'X8').
# 2. Fjern rækker uden gyldige værdier i 'X3', 'X4' eller 'X5'.
# 3. Filtrér data, så kun VFF-kampe og hjemmekampe inkluderes.

superstats <- purrr::map_dfr(2013:2023, hent_superstats_data) |>
  dplyr::select(-X1, -X6, -X7, -X8) |>  # Fjern kolonner, der ikke er nødvendige
  filter(!(X3 == "" & X4 == "" & X5 == ""))  # Fjern rækker uden værdier i relevante kolon
```

```
Har hentet data fra sæsonen: 2013/2014

Har hentet data fra sæsonen: 2014/2015

Har hentet data fra sæsonen: 2015/2016

Har hentet data fra sæsonen: 2016/2017

Har hentet data fra sæsonen: 2017/2018

Har hentet data fra sæsonen: 2018/2019

Har hentet data fra sæsonen: 2019/2020

Har hentet data fra sæsonen: 2020/2021

Har hentet data fra sæsonen: 2021/2022
```

```
Har hentet data fra sæsonen: 2022/2023

Har hentet data fra sæsonen: 2023/2024
```

```r
# ------------------------------------------------------------------------------
# Filtrér og rengør data for VFF-hjemmekampe
# ------------------------------------------------------------------------------
superstats_vff <- superstats |>
  filter(str_detect(X3, "VFF")) |>  # Behold kun rækker, hvor kolonnen 'X3' indeholder "VF
  rename(
    dato = X2,               # Omdøb 'X2' til 'dato', som repræsenterer kampdatoen
    kamp = X3,               # Omdøb 'X3' til 'kamp', som indeholder kampnavne
    resultat = X4,           # Omdøb 'X4' til 'resultat', som indeholder kampresultater
    antal_tilskuere = X5     # Omdøb 'X5' til 'antal_tilskuere', som angiver antal tilskue
  ) |>
  filter(str_starts(kamp, "VFF")) |>  # Behold kun rækker, hvor 'kamp' starter med "VFF" (
  mutate(
    dato = as.Date(dato, format = "%d/%m"),  # Konverter dato fra tekstformat (dd/mm) til
    måned = as.numeric(format(dato, "%m")),  # Uddrag månedsnummer fra dato
    årstal = ifelse(                         # Bestem årstal baseret på måneden
      måned >= 7,                            # Hvis måneden er juli eller senere
      as.numeric(substr(sæson, 1, 4)),       # Brug første år i sæsonen
      as.numeric(substr(sæson, 6, 9))        # Ellers brug andet år i sæsonen
    ),
    dato = as.Date(paste(årstal, format(dato, "%m-%d"), sep = "-")),  # Kombinér årstal, m
    antal_tilskuere = as.numeric(str_replace_all(antal_tilskuere, "\\.", ""))  # Fjern punk
  ) |>
  dplyr::select(-måned, -årstal)  # Fjern midlertidige kolonner, der ikke længere er nødve
```

```r
# ------------------------------------------------------------------------
# Tjek datastrukturen
# ------------------------------------------------------------------------
# Vi bruger glimpse til at få et overblik over datasættet og sikre, at det ser ud, som vi
glimpse(superstats_vff)
```

```
Rows: 98
Columns: 5
$ dato            <date> 2013-07-20, 2013-08-09, 2013-08-25, 2013-09-01, 2013-~
$ kamp            <chr> "VFF-RFC", "VFF-BIF", "VFF-EFB", "VFF-FCK", "VFF-FCV",~
$ resultat        <chr> "2-2", "2-2", "3-1", "1-4", "2-0", "4-1", "2-3", "2-2"~
$ antal_tilskuere <dbl> 4771, 9047, 5166, 8212, 4532, 4221, 7563, 3478, 4546, ~
$ sæson           <chr> "2013/2014", "2013/2014", "2013/2014", "2013/2014", "2~
```

```r
# ------------------------------------------------------------------------
# Læs og rengør DMI-data
# ------------------------------------------------------------------------

# ------------------------------------------------------------------------
# Opret en vektor med kampdatoer for hjemmekampe
# ------------------------------------------------------------------------

# Vi laver en vektor, der indeholder datoerne for hjemmekampe i formatet "YYYY-MM-DD".
# Dette format bruges, fordi det er påkrævet af DMI's API for at hente data.
hjemmekampe_datoer <- format(superstats_vff$dato, "%Y-%m-%d")

# ------------------------------------------------------------------------
# Basisinformation til DMI API
# ------------------------------------------------------------------------
```

```r
# Vi definerer basisinformation for API-kald til DMI:
# - base_url: URL'en for API'ets endpoint.
# - info_url: Endpointet, der bruges til at hente meteorologiske observationer.
# - station_id: Identifikatoren for den vejrstation, vi ønsker data fra.
# - api_key: API-nøglen, der giver adgang til data.

base_url <- "https://dmigw.govcloud.dk/v2/"
info_url <- "metObs/collections/observation/items?"
station_id <- "stationId=06060"  # Identifikator for vejrstationen
api_key <- "&api-key=8c762424-f683-4ad0-acea-197dc19ff258"  # API-nøgle

# ------------------------------------------------------------------------
# Funktion til at konstruere forespørgsels-URL
# ------------------------------------------------------------------------
# Denne funktion genererer en fuldstændig URL, der kan bruges til at hente data fra API'et
lav_dmi_url <- function(dato) {
  paste0(
    base_url, info_url, station_id,          # Basisinformation om API'et
    "&datetime=", dato, "T16:00:00Z/", dato, "T16:00:00Z",  # Tidsramme for observationer
    "&limit=100000", api_key                 # Begrænsning på antal rækker og API-nøgle
  )
}

# ------------------------------------------------------------------------
# Funktion til at hente data for én dato via API
# ------------------------------------------------------------------------
# Denne funktion udfører et API-kald for en specifik dato og returnerer dataen, hvis tilgæ
hent_dmi_data <- function(dato) {
```

```
  fuld_url <- lav_dmi_url(dato)  # Generér URL for datoen
  api_call <- httr::GET(fuld_url, httr::timeout(10))  # Udfør API-kald med timeout på 10 s

  if (httr::status_code(api_call) == 200) {  # Succesfuldt API-kald
    api_json <- httr::content(api_call, as = "text", encoding = "UTF-8") |>
      jsonlite::fromJSON(flatten = TRUE)  # Parse JSON-data

    if (!is.null(api_json$features) && length(api_json$features) > 0) {  # Data findes
      message("Data hentet succesfuldt for dato: ", dato)
      return(api_json$features)  # Returner data
    } else {  # Ingen data for datoen
      message("Ingen data for dato: ", dato)
      return(NULL)
    }
  } else {  # Fejl i API-kald
    message("Fejl i API-kald for dato: ", dato, " - Statuskode: ", httr::status_code(api_c
    return(NULL)
  }
}

# ------------------------------------------------------------------------------
# Hent data for alle hjemmekampe
# ------------------------------------------------------------------------------
# Iterér over alle datoer og hent data for hver kampdato via API'et.
dmi_data <- purrr::map(hjemmekampe_datoer, hent_dmi_data)
```

```
Data hentet succesfuldt for dato: 2013-07-20
```

```
Data hentet succesfuldt for dato: 2013-08-09
```

```
Data hentet succesfuldt for dato: 2013-08-25

Data hentet succesfuldt for dato: 2013-09-01

Data hentet succesfuldt for dato: 2013-09-27

Data hentet succesfuldt for dato: 2013-10-25

Data hentet succesfuldt for dato: 2013-11-08

Data hentet succesfuldt for dato: 2013-11-30

Data hentet succesfuldt for dato: 2014-02-23

Data hentet succesfuldt for dato: 2014-03-09

Data hentet succesfuldt for dato: 2014-03-30

Data hentet succesfuldt for dato: 2014-04-06

Data hentet succesfuldt for dato: 2014-04-16

Data hentet succesfuldt for dato: 2014-04-27

Data hentet succesfuldt for dato: 2014-05-08

Data hentet succesfuldt for dato: 2014-05-11

Data hentet succesfuldt for dato: 2015-07-27
```

```
Data hentet succesfuldt for dato: 2015-07-31

Data hentet succesfuldt for dato: 2015-08-16

Data hentet succesfuldt for dato: 2015-09-12

Data hentet succesfuldt for dato: 2015-09-25

Data hentet succesfuldt for dato: 2015-10-18

Data hentet succesfuldt for dato: 2015-10-24

Data hentet succesfuldt for dato: 2015-11-08

Data hentet succesfuldt for dato: 2015-11-27

Data hentet succesfuldt for dato: 2016-02-28

Data hentet succesfuldt for dato: 2016-03-12

Data hentet succesfuldt for dato: 2016-04-08

Data hentet succesfuldt for dato: 2016-04-24

Data hentet succesfuldt for dato: 2016-05-08

Data hentet succesfuldt for dato: 2016-05-16

Data hentet succesfuldt for dato: 2016-05-29
```

```
Data hentet succesfuldt for dato: 2016-07-15

Data hentet succesfuldt for dato: 2016-07-29

Data hentet succesfuldt for dato: 2016-08-12

Data hentet succesfuldt for dato: 2016-08-28

Data hentet succesfuldt for dato: 2016-09-10

Data hentet succesfuldt for dato: 2016-09-25

Data hentet succesfuldt for dato: 2016-10-02

Data hentet succesfuldt for dato: 2016-10-21

Data hentet succesfuldt for dato: 2016-11-06

Data hentet succesfuldt for dato: 2016-11-26

Data hentet succesfuldt for dato: 2016-12-12

Data hentet succesfuldt for dato: 2017-02-27

Data hentet succesfuldt for dato: 2017-03-19

Data hentet succesfuldt for dato: 2017-04-02

Data hentet succesfuldt for dato: 2017-04-17
```

```
Data hentet succesfuldt for dato: 2017-04-30

Data hentet succesfuldt for dato: 2017-05-12

Data hentet succesfuldt for dato: 2017-05-22

Data hentet succesfuldt for dato: 2021-07-31

Data hentet succesfuldt for dato: 2021-08-13

Data hentet succesfuldt for dato: 2021-08-22

Data hentet succesfuldt for dato: 2021-09-12

Data hentet succesfuldt for dato: 2021-09-26

Data hentet succesfuldt for dato: 2021-10-15

Data hentet succesfuldt for dato: 2021-10-31

Data hentet succesfuldt for dato: 2021-11-05

Data hentet succesfuldt for dato: 2021-11-28

Data hentet succesfuldt for dato: 2022-02-27

Data hentet succesfuldt for dato: 2022-03-13

Data hentet succesfuldt for dato: 2022-04-01
```

```
Data hentet succesfuldt for dato: 2022-04-17

Data hentet succesfuldt for dato: 2022-04-29

Data hentet succesfuldt for dato: 2022-05-09

Data hentet succesfuldt for dato: 2022-05-15

Data hentet succesfuldt for dato: 2022-07-17

Data hentet succesfuldt for dato: 2022-07-31

Data hentet succesfuldt for dato: 2022-08-14

Data hentet succesfuldt for dato: 2022-08-28

Data hentet succesfuldt for dato: 2022-09-12

Data hentet succesfuldt for dato: 2022-09-18

Data hentet succesfuldt for dato: 2022-10-16

Data hentet succesfuldt for dato: 2022-11-06

Data hentet succesfuldt for dato: 2023-02-20

Data hentet succesfuldt for dato: 2023-03-05

Data hentet succesfuldt for dato: 2023-03-12
```

```
Data hentet succesfuldt for dato: 2023-04-09

Data hentet succesfuldt for dato: 2023-04-23

Data hentet succesfuldt for dato: 2023-04-30

Data hentet succesfuldt for dato: 2023-05-21

Data hentet succesfuldt for dato: 2023-05-29

Data hentet succesfuldt for dato: 2023-07-28

Data hentet succesfuldt for dato: 2023-08-14

Data hentet succesfuldt for dato: 2023-08-21

Data hentet succesfuldt for dato: 2023-09-15

Data hentet succesfuldt for dato: 2023-10-01

Data hentet succesfuldt for dato: 2023-10-22

Data hentet succesfuldt for dato: 2023-11-05

Data hentet succesfuldt for dato: 2023-11-25

Data hentet succesfuldt for dato: 2024-02-16

Data hentet succesfuldt for dato: 2024-03-03
```

```
Data hentet succesfuldt for dato: 2024-03-10

Data hentet succesfuldt for dato: 2024-03-31

Data hentet succesfuldt for dato: 2024-04-21

Data hentet succesfuldt for dato: 2024-04-28

Data hentet succesfuldt for dato: 2024-05-12

Data hentet succesfuldt for dato: 2024-05-25
```

```r
names(dmi_data) <- hjemmekampe_datoer  # Navngiv listen med kampdatoer


# ------------------------------------------------------------------------------
# Kombiner og rengør data fra DMI
# ------------------------------------------------------------------------------
# Omstrukturer data til bredt format og fjern irrelevante kolonner.
dmi_variabler <- purrr::map_dfr(names(dmi_data), function(dato) {
  if (!is.null(dmi_data[[dato]])) {
    dmi_data[[dato]] |>
      as_tibble() |>
      dplyr::select(
        Observationstidspunkt = properties.observed,
        Observationer = properties.parameterId,
        Værdi = properties.value
      ) |>
      mutate(
        Observationstidspunkt = lubridate::ymd_hms(Observationstidspunkt),  # Konverter ti
```

```
        dato = as.Date(dato)  # Tilføj kampdato
      )
  }
}) |>
  tidyr::pivot_wider(names_from = Observationer, values_from = Værdi) |>
  dplyr::select(-Observationstidspunkt)

# ------------------------------------------------------------------------
# Fjern irrelevante variabler
# ------------------------------------------------------------------------
# Vi fjerner kolonner, der indeholder minimum, maksimum, eller midlingsværdier, da de ikke
dmi_variabler <- dmi_variabler |>
  dplyr::select(-contains("min"), -contains("max"), -contains("1h"), -contains("mean"))

# ------------------------------------------------------------------------
# Håndtering af variabler med mange manglende værdier
# ------------------------------------------------------------------------
# Kolonnen 'weather' indeholder mange NA-værdier, men vi kan stadig udlede nyttig informat
# Vi opretter en dummy-variabel for nedbør baseret på API-dokumentation.
dmi_variabler <- dmi_variabler |>
  mutate(
    nedbør = if_else(
      weather %in% c(121, 123, 124, 125, 126, 140:149, 150:159, 160:169, 170:179, 180:189,
      1,  # Nedbør til stede
      0   # Ingen nedbør
    )
  ) |>
  dplyr::select(-weather)  # Fjern den oprindelige 'weather'-kolonne
```

```r
# ------------------------------------------------------------------------------
# Udfyld manglende værdier med medianer
# ------------------------------------------------------------------------------
# For at sikre datasættets konsistens udfylder vi alle manglende værdier med kolonne-media
dmi_variabler <- dmi_variabler |>
  mutate(across(where(is.numeric), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))

# ------------------------------------------------------------------------------
# Kontroller datastruktur
# ------------------------------------------------------------------------------
# Vi bruger glimpse til at få et overblik over datasættet og sikre, at det er i korrekt fo
glimpse(dmi_variabler)
```

```
Rows: 98
Columns: 13
$ dato            <date> 2013-07-20, 2013-08-09, 2013-08-25, 2013-09-01, 2013-~
$ temp_dew        <dbl> 12.9, 13.0, 10.4, 11.4, 5.2, 9.0, 5.7, 1.6, 4.6, 4.5, ~
$ pressure_at_sea <dbl> 1022.6, 1017.4, 1020.8, 1015.1, 1017.0, 1008.2, 1005.2~
$ temp_grass      <dbl> 32.9, 22.1, 25.9, 11.8, 14.5, 10.1, 5.6, 0.3, 7.0, 12.~
$ wind_dir        <dbl> 240, 247, 82, 278, 347, 134, 188, 288, 199, 221, 358, ~
$ pressure        <dbl> 1016.5, 1011.2, 1014.6, 1008.7, 1010.6, 1001.8, 998.8,~
$ humidity        <dbl> 47, 62, 48, 93, 59, 86, 87, 95, 79, 56, 73, 97, 41, 51~
$ wind_speed      <dbl> 5.1, 4.6, 4.1, 6.7, 3.6, 5.7, 2.6, 3.6, 7.7, 6.7, 3.1,~
$ temp_dry        <dbl> 24.9, 20.5, 21.8, 12.5, 13.1, 11.4, 7.8, 2.3, 8.0, 13.~
$ visibility      <dbl> 35000, 35000, 35000, 2900, 35000, 35000, 35000, 35000,~
$ cloud_cover     <dbl> 90, 90, 90, 90, 90, 90, 90, 90, 75, 0, 0, 100, 90, 0, ~
$ cloud_height    <dbl> 800, 800, 800, 800, 800, 800, 800, 800, 1250, 800, 800~
$ nedbør          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, ~
```

```r
# ------------------------------------------------------------------------------
# Dokumentation for Meterological Observation Data
# ------------------------------------------------------------------------------
# Dokumentation for variabler kan findes her:
# https://opendatadocs.dmi.govcloud.dk/en/Data/Meteorological_Observation_Data

# ------------------------------------------------------------------------------
# Vi samler de tre datasæt
# ------------------------------------------------------------------------------

# Her samler vi data fra de tre tidligere forberedte datasæt:
# 1. `excel_guld`: Indeholder oplysninger om VIP-guldmenuer.
# 2. `superstats_vff`: Indeholder kampdata for VFF's hjemmekampe.
# 3. `dmi_variabler`: Indeholder meteorologiske observationer for kampdage.

# Ved at bruge `inner_join` sikrer vi, at kun rækker med matchende `dato` i alle tre
# datasæt inkluderes i det endelige datasæt `joins_samlet`.

joins_samlet <- excel_guld |>
  inner_join(superstats_vff, by = "dato") |>  # Sammenkæd Excel-data med Superstats-data ba
  inner_join(dmi_variabler, by = "dato")      # Tilføj DMI-data ved at matche på `dato`.

# ------------------------------------------------------------------------------
# Tjek datasættet for struktur og NA-værdier
# ------------------------------------------------------------------------------

# Vi bruger `glimpse` for at få et hurtigt overblik over datastrukturen.
# Dette giver os indsigt i variabelnavne, datatyper og eksempler på værdier.
```

```r
glimpse(joins_samlet)
```

```
Rows: 98
Columns: 18
$ dato            <date> 2013-07-20, 2013-08-09, 2013-08-25, 2013-09-01, 2013-~
$ guld_menu_stk   <dbl> 470, 791, 644, 768, 682, 748, 682, 514, 412, 535, 680,~
$ kamp            <chr> "VFF-RFC", "VFF-BIF", "VFF-EFB", "VFF-FCK", "VFF-FCV",~
$ resultat        <chr> "2-2", "2-2", "3-1", "1-4", "2-0", "4-1", "2-3", "2-2"~
$ antal_tilskuere <dbl> 4771, 9047, 5166, 8212, 4532, 4221, 7563, 3478, 4546, ~
$ sæson           <chr> "2013/2014", "2013/2014", "2013/2014", "2013/2014", "2~
$ temp_dew        <dbl> 12.9, 13.0, 10.4, 11.4, 5.2, 9.0, 5.7, 1.6, 4.6, 4.5, ~
$ pressure_at_sea <dbl> 1022.6, 1017.4, 1020.8, 1015.1, 1017.0, 1008.2, 1005.2~
$ temp_grass      <dbl> 32.9, 22.1, 25.9, 11.8, 14.5, 10.1, 5.6, 0.3, 7.0, 12.~
$ wind_dir        <dbl> 240, 247, 82, 278, 347, 134, 188, 288, 199, 221, 358, ~
$ pressure        <dbl> 1016.5, 1011.2, 1014.6, 1008.7, 1010.6, 1001.8, 998.8,~
$ humidity        <dbl> 47, 62, 48, 93, 59, 86, 87, 95, 79, 56, 73, 97, 41, 51~
$ wind_speed      <dbl> 5.1, 4.6, 4.1, 6.7, 3.6, 5.7, 2.6, 3.6, 7.7, 6.7, 3.1,~
$ temp_dry        <dbl> 24.9, 20.5, 21.8, 12.5, 13.1, 11.4, 7.8, 2.3, 8.0, 13.~
$ visibility      <dbl> 35000, 35000, 35000, 2900, 35000, 35000, 35000, 35000,~
$ cloud_cover     <dbl> 90, 90, 90, 90, 90, 90, 90, 90, 75, 0, 0, 100, 90, 0, ~
$ cloud_height    <dbl> 800, 800, 800, 800, 800, 800, 800, 800, 1250, 800, 800~
$ nedbør          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, ~
```

```r
# `colSums(is.na(joins_samlet))` giver en oversigt over, hvor mange manglende værdier
# der er i hver kolonne. Dette hjælper med at identificere variabler, der skal håndteres
# yderligere (fx imputering af manglende værdier eller fjernelse af variabler).
colSums(is.na(joins_samlet))
```

```
          dato   guld_menu_stk            kamp        resultat antal_tilskuere
```

```
            0             0             0             0             0
       sæson      temp_dew pressure_at_sea    temp_grass      wind_dir
            0             0             0             0             0
    pressure      humidity    wind_speed      temp_dry    visibility
            0             0             0             0             0
cloud_cover  cloud_height        nedbør
            0             0             0
```

```r
# Ved at kombinere disse trin sikrer vi, at det samlede datasæt er klart til
# yderligere analyse og modellering.

#==============================================================================
#                          2. Oprettelse af nye variabler
#==============================================================================

# ------------------------------------------------------------------------------
# Tilføj ugedag og årstider
# ------------------------------------------------------------------------------
# Vi tilføjer variabler, der repræsenterer ugedagen og årstiden for hver kampdato.
# Dette kan bruges til at identificere mønstre i data, som fx hvilke ugedage eller
# årstider der har flere VIP-guldmenuer.

joins_samlet_variabler <- joins_samlet |>
  mutate(
    ugedag = as.factor(weekdays(dato)),  # Konverter kampdato til ugedag som kategorisk va
    aarstid = case_when(  # Opdel kampdato i årstider baseret på månedsnummer.
      month(dato) %in% c(12, 1, 2) ~ "Vinter",
      month(dato) %in% c(3, 4, 5) ~ "Forår",
      month(dato) %in% c(6, 7, 8) ~ "Sommer",
```

```
      month(dato) %in% c(9, 10, 11) ~ "Efterår"
    ) |> factor(levels = c("Vinter", "Forår", "Sommer", "Efterår"))  # Sørg for logisk rækk
  ) |>
  relocate(ugedag, .after = dato) |>  # Flyt 'ugedag' tættere på 'dato' for bedre overskue
  relocate(aarstid, .after = ugedag)  # Flyt 'aarstid' lige efter 'ugedag'.

# ------------------------------------------------------------------------
# Tilføj modstanderkategorier
# ------------------------------------------------------------------------
# Vi opretter variabler for modstanderen og dens kategori baseret på kampens modstander.
# Modstanderne kategoriseres som 'Topklub', 'Midterklub' eller 'Bundklub' afhængigt
# af deres relative styrke.

joins_samlet_variabler <- joins_samlet_variabler |>
  mutate(
    modstander = kamp |>  # Ekstraher modstanderens navn fra kampens tekst.
      str_remove(".*VFF-") |>
      str_trim() |>
      as.factor(),
    modstander_kategori = case_when(  # Kategorisér modstanderen i tre niveauer.
      modstander %in% c("FCK", "FCM", "BIF", "AGF") ~ "Topklub",
      modstander %in% c("SIF", "RFC", "FCN", "AAB") ~ "Midterklub",
      TRUE ~ "Bundklub"
    ) |> factor(levels = c("Topklub", "Midterklub", "Bundklub"))
  ) |>
  relocate(modstander, .after = kamp) |>  # Flyt 'modstander' tættere på 'kamp'.
  relocate(modstander_kategori, .after = modstander)  # Placér 'modstander_kategori' efter
```

```r
# ------------------------------------------------------------------------------
# Tilføj tilskuere fra sidste møde
# ------------------------------------------------------------------------------
# Vi tilføjer en variabel, der angiver antallet af tilskuere fra den foregående kamp
# mod samme modstander. Dette kan bruges til at analysere trends mellem kampe.

joins_samlet_variabler <- joins_samlet_variabler |>
  arrange(modstander, dato) |>  # Sortér efter modstander og kampdato for korrekt kronolog
  group_by(modstander) |>  # Gruppér data efter modstander.
  mutate(
    tilskuere_sidste_møde = lag(antal_tilskuere),  # Brug 'lag' til at finde tilskuere fra
    tilskuere_sidste_møde = ifelse(
      is.na(tilskuere_sidste_møde),
      slide_dbl(antal_tilskuere, mean, .before = 2, .complete = TRUE),  # Brug gennemsnit
      tilskuere_sidste_møde
    ),
    tilskuere_sidste_møde = ifelse(
      is.na(tilskuere_sidste_møde),
      median(antal_tilskuere, na.rm = TRUE),  # Brug medianen som fallback for at håndtere
      tilskuere_sidste_møde
    )
  ) |>
  ungroup() |>  # Fjern gruppering for at undgå utilsigtede påvirkninger.
  arrange(dato) |>  # Gendan oprindelig rækkefølge efter dato.
  relocate(tilskuere_sidste_møde, .after = antal_tilskuere)  # Placér 'tilskuere_sidste_mø

# ------------------------------------------------------------------------------
# Tilføj guldmenuer fra sidste møde
```

```r
# ------------------------------------------------------------------------------
# Vi opretter en variabel, der repræsenterer antallet af guldmenuer, der blev afhentet
# ved det sidste møde mod samme modstander.

joins_samlet_variabler <- joins_samlet_variabler |>
  arrange(modstander, dato) |>  # Sortér efter modstander og kampdato.
  group_by(modstander) |>  # Gruppér efter modstander.
  mutate(
    guld_menuer_sidste_møde = lag(guld_menu_stk),  # Brug 'lag' til at finde guldmenuer fr
    guld_menuer_sidste_møde = ifelse(
      is.na(guld_menuer_sidste_møde),
      slide_dbl(guld_menu_stk, mean, .before = 2, .complete = TRUE),  # Brug gennemsnit fr
      guld_menuer_sidste_møde
    ),
    guld_menuer_sidste_møde = ifelse(
      is.na(guld_menuer_sidste_møde),
      median(guld_menu_stk, na.rm = TRUE),  # Brug median som fallback.
      guld_menuer_sidste_møde
    )
  ) |>
  ungroup() |>  # Fjern gruppering.
  arrange(dato) |>  # Gendan dato-rækkefølge.
  relocate(guld_menuer_sidste_møde, .after = guld_menu_stk)  # Placér variablen lige efter

# ------------------------------------------------------------------------------
# Inddel modstanderne i kategorier ud fra antal af guldmenuer
# ------------------------------------------------------------------------------
# Vi opretter en kategori baseret på antallet af guldmenuer, der afhentes mod
```

```r
# forskellige modstandere.

joins_samlet_variabler <- joins_samlet_variabler |>
  mutate(
    guld_menu_kategori = case_when(
      modstander %in% c("HIF", "VB") ~ "A",  # Modstandere med lavere antal guldmenuer.
      modstander %in% c("BIF", "OB", "FCM", "AGF") ~ "B",  # Modstandere med middel antal.
      TRUE ~ "C"  # Modstandere med højere antal guldmenuer.
    )
  ) |>
  relocate(guld_menu_kategori, .after = guld_menuer_sidste_møde)  # Placér variablen efter

# ------------------------------------------------------------------------------
# Hjælpefunktion til at beregne point fra kampresultat
# ------------------------------------------------------------------------------
# Denne funktion tager kampresultatet som input (fx "2-1") og beregner point:
# - 3 point for en sejr
# - 1 point for uafgjort
# - 0 point for et nederlag
# Funktionen returnerer NA, hvis resultatet ikke er gyldigt.

parse_result <- function(result) {
  # Ekstraherer tallene fra resultatet (fx "2-1" -> c(2, 1))
  scores <- as.numeric(str_extract_all(result, "\\d+")[[1]])

  # Beregner point baseret på scoren
  case_when(
    scores[1] > scores[2] ~ 3,   # Hjemmesejr
```

```
    scores[1] == scores[2] ~ 1,  # Uafgjort
    scores[1] < scores[2] ~ 0,   # Hjemmenederlag
    TRUE ~ NA_real_              # Returnerer NA for manglende eller ugyldige data
  )
}

# ------------------------------------------------------------------------------
# Tilføj point for de seneste 3 kampe
# ------------------------------------------------------------------------------
# Vi beregner den samlede pointscore for de seneste 3 kampe og tilføjer resultatet
# som en ny kolonne, "point_sidste_3_kampe". Dette giver os et overblik over
# holdets form i de seneste kampe.

joins_samlet_variabler <- joins_samlet_variabler |>
  mutate(
    # Beregn point for de seneste 3 kampe ved at summere pointene
    point_sidste_3_kampe = rowSums(
      cbind(
        # Punkt 1: Beregn point fra den seneste kamp
        lag(resultat, 1) |> purrr::map_dbl(~ parse_result(.x)),

        # Punkt 2: Beregn point fra den næstseneste kamp
        lag(resultat, 2) |> purrr::map_dbl(~ parse_result(.x)),

        # Punkt 3: Beregn point fra den tredje seneste kamp
        lag(resultat, 3) |> purrr::map_dbl(~ parse_result(.x))
      ),
      na.rm = TRUE  # Ignorer manglende værdier (fx hvis der ikke er data for 3 kampe)
```

```
    )
  ) |>
  # Flyt den nye kolonne, så den kommer lige efter "resultat" for bedre overskuelighed
  relocate(point_sidste_3_kampe, .after = resultat)

# ------------------------------------------------------------------------------
# Kontrol af datasæt
# ------------------------------------------------------------------------------
# Vi bruger `glimpse` til at undersøge strukturen af datasættet og sikre,
# at den nye variabel "point_sidste_3_kampe" er tilføjet korrekt.
glimpse(joins_samlet_variabler)
```

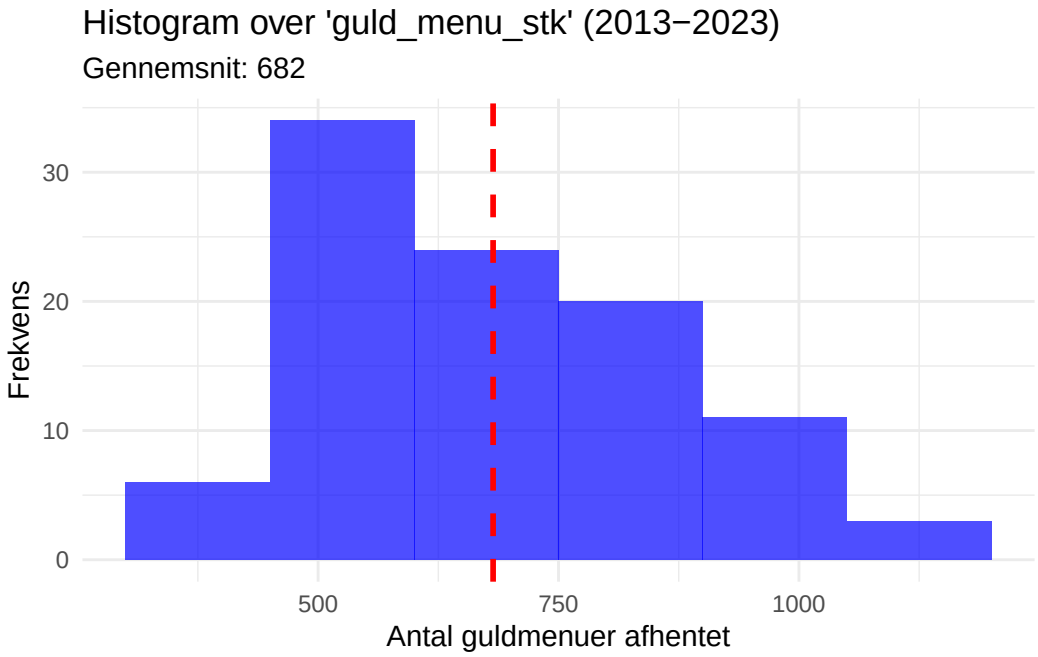
```
Rows: 98
Columns: 26
$ dato                    <date> 2013-07-20, 2013-08-09, 2013-08-25, 2013-09-0~
$ ugedag                  <fct> Saturday, Friday, Sunday, Sunday, Friday, Frid~
$ aarstid                 <fct> Sommer, Sommer, Sommer, Efterår, Efterår, Efte~
$ guld_menu_stk           <dbl> 470, 791, 644, 768, 682, 748, 682, 514, 412, 5~
$ guld_menuer_sidste_møde <dbl> 686.0, 791.0, 535.0, 765.5, 715.0, 682.0, 708.~
$ guld_menu_kategori      <chr> "C", "B", "C", "C", "C", "C", "B", "C", "C", "~
$ kamp                    <chr> "VFF-RFC", "VFF-BIF", "VFF-EFB", "VFF-FCK", "V~
$ modstander              <fct> RFC, BIF, EFB, FCK, FCV, FCV, FCM, SJF, RFC, E~
$ modstander_kategori     <fct> Midterklub, Topklub, Bundklub, Topklub, Bundkl~
$ resultat                <chr> "2-2", "2-2", "3-1", "1-4", "2-0", "4-1", "2-3~
$ point_sidste_3_kampe    <dbl> 0, 1, 2, 5, 4, 6, 6, 6, 4, 2, 2, 1, 1, 1, 1, 0~
$ antal_tilskuere         <dbl> 4771, 9047, 5166, 8212, 4532, 4221, 7563, 3478~
$ tilskuere_sidste_møde   <dbl> 5085.0, 7747.0, 3557.0, 6362.5, 4376.5, 4532.0~
$ sæson                   <chr> "2013/2014", "2013/2014", "2013/2014", "2013/2~
```

```
$ temp_dew              <dbl> 12.9, 13.0, 10.4, 11.4, 5.2, 9.0, 5.7, 1.6, 4.~
$ pressure_at_sea       <dbl> 1022.6, 1017.4, 1020.8, 1015.1, 1017.0, 1008.2~
$ temp_grass            <dbl> 32.9, 22.1, 25.9, 11.8, 14.5, 10.1, 5.6, 0.3, ~
$ wind_dir              <dbl> 240, 247, 82, 278, 347, 134, 188, 288, 199, 22~
$ pressure              <dbl> 1016.5, 1011.2, 1014.6, 1008.7, 1010.6, 1001.8~
$ humidity              <dbl> 47, 62, 48, 93, 59, 86, 87, 95, 79, 56, 73, 97~
$ wind_speed            <dbl> 5.1, 4.6, 4.1, 6.7, 3.6, 5.7, 2.6, 3.6, 7.7, 6~
$ temp_dry              <dbl> 24.9, 20.5, 21.8, 12.5, 13.1, 11.4, 7.8, 2.3, ~
$ visibility            <dbl> 35000, 35000, 35000, 2900, 35000, 35000, 35000~
$ cloud_cover           <dbl> 90, 90, 90, 90, 90, 90, 90, 90, 75, 0, 0, 100,~
$ cloud_height          <dbl> 800, 800, 800, 800, 800, 800, 800, 800, 1250, ~
$ nedbør                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1~
```

```
#==============================================================================
#                       3. Eksplorativ dataanalyse (EDA)
#==============================================================================


# ------------------------------------------------------------------------------
# Udtræk kun numeriske variabler fra `joins_samlet`
# ------------------------------------------------------------------------------
# Vi filtrerer datasættet til kun at indeholde numeriske variabler for at kunne
# udføre kvantitative analyser som korrelation og lineære modeller.
joins_samlet_numerisk <- joins_samlet_variabler |>
  dplyr::select(where(is.numeric))


# ------------------------------------------------------------------------------
# Histogram der viser hvor mange guldmenuer der afsættes 2013-2023
# ------------------------------------------------------------------------------
# Beregn gennemsnittet af guldmenuer for hele perioden.
```

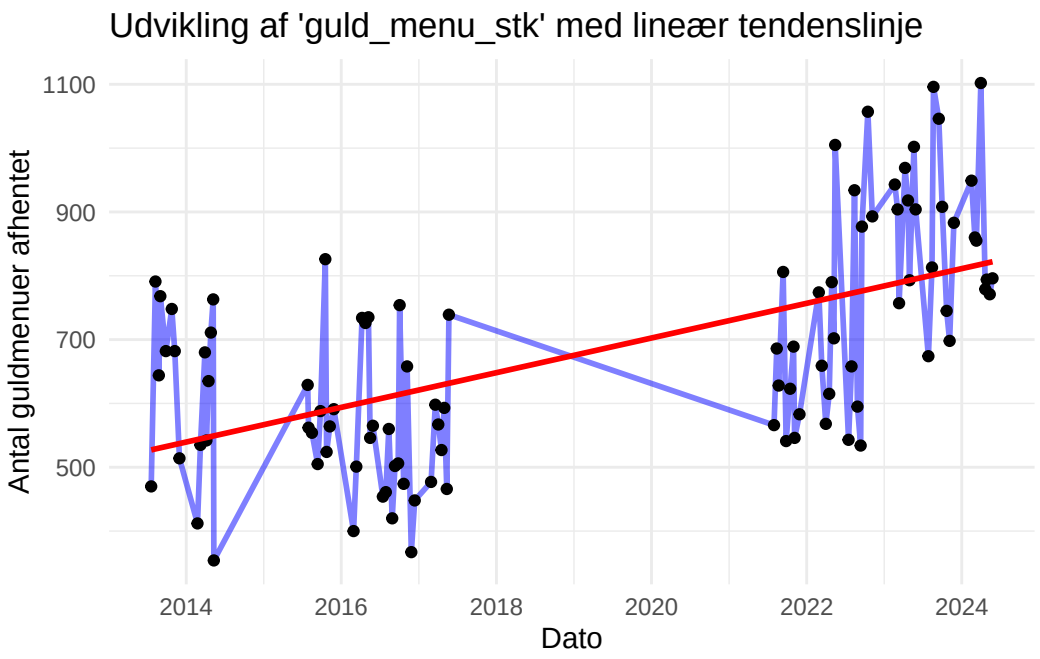
```r
gennemsnit_2013_2023 <- joins_samlet_numerisk |>
  summarise(gennemsnit = round(mean(guld_menu_stk, na.rm = TRUE))) |>
  pull(gennemsnit)

# Visualiser fordelingen af guldmenuer med et histogram og marker gennemsnittet.
joins_samlet_numerisk |>
  ggplot(aes(x = guld_menu_stk)) +
  geom_histogram(binwidth = 150, fill = "blue", alpha = 0.7, boundary = 0) +
  geom_vline(xintercept = gennemsnit_2013_2023, color = "red", linetype = "dashed", size =
  labs(
    title = "Histogram over 'guld_menu_stk' (2013-2023)",
    subtitle = paste0("Gennemsnit: ", round(gennemsnit_2013_2023, 2)),
    x = "Antal guldmenuer afhentet",
    y = "Frekvens"
  ) +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



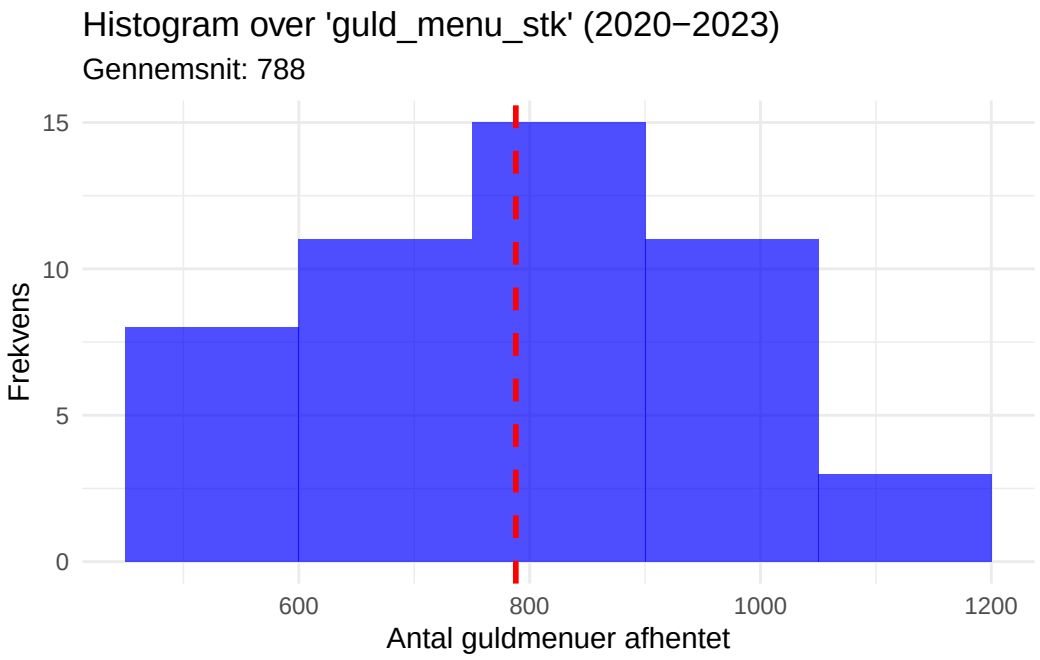

```
# ------------------------------------------------------------------------------
# Udviklingen af afsatte guldmenuer i perioden
# ------------------------------------------------------------------------------
# Visualiser udviklingen af guldmenuer over tid med en lineær tendenslinje.
joins_samlet |>
  ggplot(aes(x = dato, y = guld_menu_stk)) +
  geom_line(color = "blue", size = 1, alpha = 0.5) +
  geom_point(color = "black", size = 1.5) +
  geom_smooth(method = "lm", color = "red", se = FALSE, size = 1) +  # Tilføj lineær tende
  labs(
    title = "Udvikling af 'guld_menu_stk' med lineær tendenslinje",
    x = "Dato",
    y = "Antal guldmenuer afhentet"
  ) +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



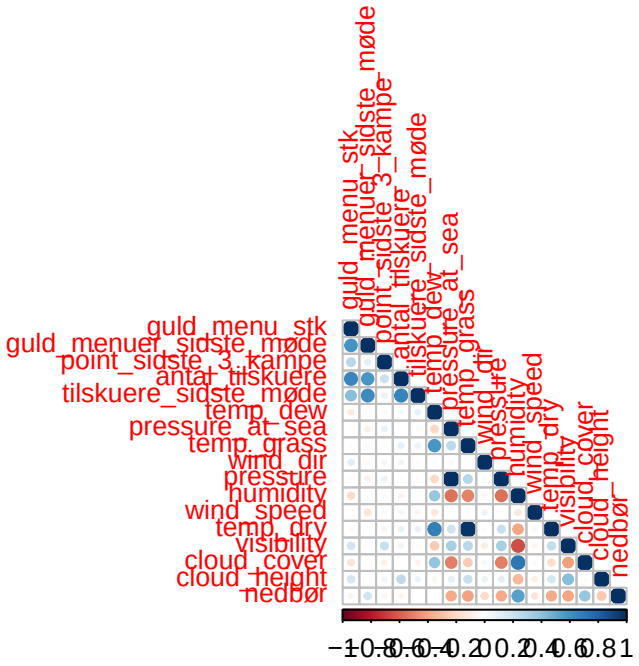

```
# ------------------------------------------------------------------------------
# Histogram der viser hvor mange guldmenuer der afsættes 2020-2023
# ------------------------------------------------------------------------------
# Filtrér data for perioden 2020/2021 og frem.
joins_samlet_2020_frem <- joins_samlet |>
  filter(dato >= as.Date("2020-07-01"))

# Beregn gennemsnittet for denne nyere periode.
gennemsnit_2020_frem <- joins_samlet_2020_frem |>
  summarise(gennemsnit = round(mean(guld_menu_stk, na.rm = TRUE))) |>
  pull(gennemsnit)

# Visualiser data for denne periode med et histogram og marker gennemsnittet.
joins_samlet_2020_frem |>
```

```r
  ggplot(aes(x = guld_menu_stk)) +
  geom_histogram(binwidth = 150, fill = "blue", alpha = 0.7, boundary = 0) +
  geom_vline(xintercept = gennemsnit_2020_frem, color = "red", linetype = "dashed", size =
  labs(
    title = "Histogram over 'guld_menu_stk' (2020-2023)",
    subtitle = paste0("Gennemsnit: ", round(gennemsnit_2020_frem, 2)),
    x = "Antal guldmenuer afhentet",
    y = "Frekvens"
  ) +
  theme_minimal()
```




```r
# ------------------------------------------------------------------------------
# Korrellationsmatrix
# ------------------------------------------------------------------------------
# Beregn korrelationsmatrix for numeriske variabler og afrund værdierne.
cor_matrix <- cor(joins_samlet_numerisk, use = "pairwise.complete.obs")
```

```
cor_matrix_rounded <- round(cor_matrix, 2)


# Visualiser korrelationer mellem variabler.
corrplot(cor_matrix, method = "circle", type = "lower", tl.cex = 0.8)
```




```
# Fjern stærkt korrelerede variabler, som kan medføre multikollinearitet.
joins_samlet_numerisk <- joins_samlet_numerisk |>
  dplyr::select(-pressure_at_sea, -temp_grass, -temp_dew)


# Gem listen over fjernede variabler til fremtidig reference.
stærkt_korrelerede_variabler <- c("pressure_at_sea", "temp_grass", "temp_dew")


# ------------------------------------------------------------------------------
# Lav en lineær model med numeriske variabler og beregn VIF
# ------------------------------------------------------------------------------
# Byg en lineær model for at undersøge multikollinearitet med Variance Inflation Factor (V
```

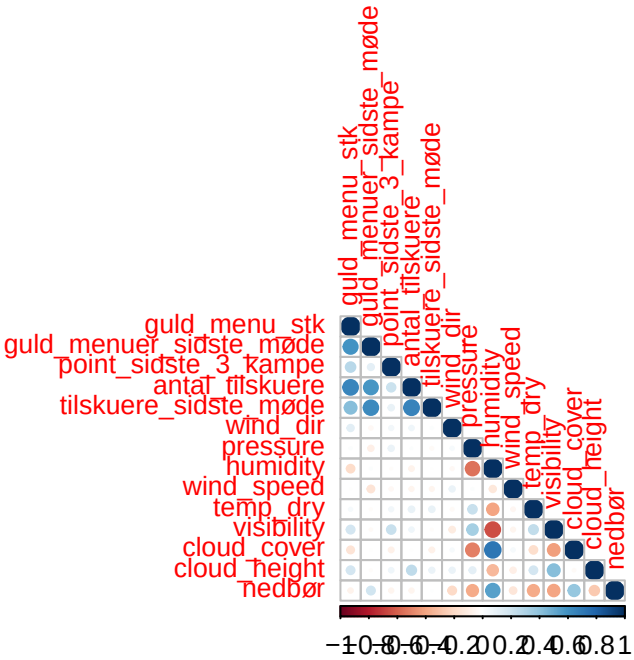
```r
model <- lm(guld_menu_stk ~ ., data = joins_samlet_numerisk)
vif_values <- vif(model)  # Beregn VIF-værdier.

# Identificér variabler med høj VIF, der kan indikere redundans.
high_vif <- vif_values[vif_values > 5]
print("Variabler med høj VIF (over 5):")
```

```
[1] "Variabler med høj VIF (over 5):"
```

```r
print(high_vif)
```

```
humidity
5.047272
```

```r
# Vi er opmærksomme på humidity, men bevidste om at Ridge og Lasso kan håndtere variabler m

# ------------------------------------------------------------------------
# Visualisering af ny korrelationsmatrix efter fjernelse af korrelerede variabler
# ------------------------------------------------------------------------
cor_matrix <- cor(joins_samlet_numerisk, use = "pairwise.complete.obs")
cor_matrix_rounded <- round(cor_matrix, 2)

# Visualiser opdateret korrelationsmatrix.
corrplot(cor_matrix, method = "circle", type = "lower", tl.cex = 0.8)
```




```
# ------------------------------------------------------------------------------
# Kontrol af datasættets struktur
# ------------------------------------------------------------------------------
# Tjek datasættet for at sikre, at de ønskede ændringer er blevet anvendt korrekt.
glimpse(joins_samlet_variabler)
```

```
Rows: 98
Columns: 26
$ dato                    <date> 2013-07-20, 2013-08-09, 2013-08-25, 2013-09-0~
$ ugedag                  <fct> Saturday, Friday, Sunday, Sunday, Friday, Frid~
$ aarstid                 <fct> Sommer, Sommer, Sommer, Efterår, Efterår, Efte~
$ guld_menu_stk           <dbl> 470, 791, 644, 768, 682, 748, 682, 514, 412, 5~
$ guld_menuer_sidste_møde <dbl> 686.0, 791.0, 535.0, 765.5, 715.0, 682.0, 708.~
$ guld_menu_kategori      <chr> "C", "B", "C", "C", "C", "C", "B", "C", "C", "~
$ kamp                    <chr> "VFF-RFC", "VFF-BIF", "VFF-EFB", "VFF-FCK", "V~
$ modstander              <fct> RFC, BIF, EFB, FCK, FCV, FCV, FCM, SJF, RFC, E~
```

```
$ modstander_kategori     <fct> Midterklub, Topklub, Bundklub, Topklub, Bundkl~
$ resultat                <chr> "2-2", "2-2", "3-1", "1-4", "2-0", "4-1", "2-3~
$ point_sidste_3_kampe    <dbl> 0, 1, 2, 5, 4, 6, 6, 6, 4, 2, 2, 1, 1, 1, 1, 0~
$ antal_tilskuere         <dbl> 4771, 9047, 5166, 8212, 4532, 4221, 7563, 3478~
$ tilskuere_sidste_møde   <dbl> 5085.0, 7747.0, 3557.0, 6362.5, 4376.5, 4532.0~
$ sæson                   <chr> "2013/2014", "2013/2014", "2013/2014", "2013/2~
$ temp_dew                <dbl> 12.9, 13.0, 10.4, 11.4, 5.2, 9.0, 5.7, 1.6, 4.~
$ pressure_at_sea         <dbl> 1022.6, 1017.4, 1020.8, 1015.1, 1017.0, 1008.2~
$ temp_grass              <dbl> 32.9, 22.1, 25.9, 11.8, 14.5, 10.1, 5.6, 0.3, ~
$ wind_dir                <dbl> 240, 247, 82, 278, 347, 134, 188, 288, 199, 22~
$ pressure                <dbl> 1016.5, 1011.2, 1014.6, 1008.7, 1010.6, 1001.8~
$ humidity                <dbl> 47, 62, 48, 93, 59, 86, 87, 95, 79, 56, 73, 97~
$ wind_speed              <dbl> 5.1, 4.6, 4.1, 6.7, 3.6, 5.7, 2.6, 3.6, 7.7, 6~
$ temp_dry                <dbl> 24.9, 20.5, 21.8, 12.5, 13.1, 11.4, 7.8, 2.3, ~
$ visibility              <dbl> 35000, 35000, 35000, 2900, 35000, 35000, 35000~
$ cloud_cover             <dbl> 90, 90, 90, 90, 90, 90, 90, 90, 75, 0, 0, 100,~
$ cloud_height            <dbl> 800, 800, 800, 800, 800, 800, 800, 800, 1250, ~
$ nedbør                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1~
```

```
#==============================================================================
#                  4. Forberedelse af data til modellering
#==============================================================================


# ------------------------------------------------------------------------------
# Fjernelse af irrelevante variabler
# ------------------------------------------------------------------------------
# Vi fjerner variabler, der ikke er nødvendige for vores analyse eller modellering.
# Dette inkluderer dato, kamp, modstander, sæson, resultat og stærkt korrelerede variabler
# som vi tidligere har identificeret og lagret i `stærkt_korrelerede_variabler`.
```

```r
joins_samlet_model <- joins_samlet_variabler |>
  dplyr::select(
    -dato,                              # Fjern dato, da den ikke direkte bidrager til mod
    -kamp,                              # Fjern kampnavn, da det ikke har numerisk eller k
    -modstander,                        # Fjern modstandernavn, da vi har kategoriseret de
    -sæson,                             # Fjern sæson, da det ikke er relevant for modelle
    -resultat,                          # Fjern kampresultat, da vi allerede har udledt po
    -all_of(stærkt_korrelerede_variabler)  # Fjern stærkt korrelerede variabler for at red
  )

# Tjek datasættets struktur og de resterende variabler
glimpse(joins_samlet_model)
```

```
Rows: 98
Columns: 18
$ ugedag                  <fct> Saturday, Friday, Sunday, Sunday, Friday, Frid~
$ aarstid                 <fct> Sommer, Sommer, Sommer, Efterår, Efterår, Efte~
$ guld_menu_stk           <dbl> 470, 791, 644, 768, 682, 748, 682, 514, 412, 5~
$ guld_menuer_sidste_møde <dbl> 686.0, 791.0, 535.0, 765.5, 715.0, 682.0, 708.~
$ guld_menu_kategori      <chr> "C", "B", "C", "C", "C", "C", "B", "C", "C", "~
$ modstander_kategori     <fct> Midterklub, Topklub, Bundklub, Topklub, Bundkl~
$ point_sidste_3_kampe    <dbl> 0, 1, 2, 5, 4, 6, 6, 6, 4, 2, 2, 1, 1, 1, 1, 0~
$ antal_tilskuere         <dbl> 4771, 9047, 5166, 8212, 4532, 4221, 7563, 3478~
$ tilskuere_sidste_møde   <dbl> 5085.0, 7747.0, 3557.0, 6362.5, 4376.5, 4532.0~
$ wind_dir                <dbl> 240, 247, 82, 278, 347, 134, 188, 288, 199, 22~
$ pressure                <dbl> 1016.5, 1011.2, 1014.6, 1008.7, 1010.6, 1001.8~
$ humidity                <dbl> 47, 62, 48, 93, 59, 86, 87, 95, 79, 56, 73, 97~
$ wind_speed              <dbl> 5.1, 4.6, 4.1, 6.7, 3.6, 5.7, 2.6, 3.6, 7.7, 6~
```

```
$ temp_dry                <dbl> 24.9, 20.5, 21.8, 12.5, 13.1, 11.4, 7.8, 2.3, ~
$ visibility              <dbl> 35000, 35000, 35000, 2900, 35000, 35000, 35000~
$ cloud_cover             <dbl> 90, 90, 90, 90, 90, 90, 90, 90, 75, 0, 0, 100,~
$ cloud_height            <dbl> 800, 800, 800, 800, 800, 800, 800, 800, 1250, ~
$ nedbør                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1~
```

```
# ------------------------------------------------------------------------------
# Omdøbning af kolonner for bedre læsbarhed
# ------------------------------------------------------------------------------
# Vi ændrer kolonnenavne til mere beskrivende og letforståelige danske navne.
joins_samlet_model <- joins_samlet_model |>
  rename(
    vindretning = wind_dir,             # Vindretning
    lufttryk = pressure,                # Lufttryk
    luftfugtighed = humidity,           # Luftfugtighed
    vindhastighed = wind_speed,         # Vindhastighed
    lufttemperatur = temp_dry,          # Lufttemperatur
    sigtbarhed = visibility,            # Sigtbarhed
    skydække = cloud_cover,             # Skydække
    skyhøjde = cloud_height             # Skyhøjde
  )


# ------------------------------------------------------------------------------
# Konvertering af datatyper
# ------------------------------------------------------------------------------
# Vi sikrer, at variablerne har de rigtige datatyper (numerisk eller kategorisk),
# som krævet for modellering og analyse.
joins_samlet_model <- joins_samlet_model |>
  mutate(
```

```r
    ugedag = as.factor(ugedag),                    # Konverter ugedag til kategorisk variab
    aarstid = as.factor(aarstid),                  # Konverter årstid til kategorisk variab
    guld_menu_stk = as.numeric(guld_menu_stk),     # Konverter til numerisk format.
    guld_menuer_sidste_møde = as.numeric(guld_menuer_sidste_møde),  # Konverter til numeri
    guld_menu_kategori = as.factor(guld_menu_kategori),  # Konverter til kategorisk variab
    modstander_kategori = as.factor(modstander_kategori),  # Konverter til kategorisk vari
    point_sidste_3_kampe = as.numeric(point_sidste_3_kampe),  # Konverter til numerisk for
    antal_tilskuere = as.numeric(antal_tilskuere),  # Konverter til numerisk format.
    tilskuere_sidste_møde = as.numeric(tilskuere_sidste_møde),  # Konverter til numerisk f
    vindretning = as.numeric(vindretning),         # Konverter til numerisk format.
    lufttryk = as.numeric(lufttryk),               # Konverter til numerisk format.
    luftfugtighed = as.numeric(luftfugtighed),     # Konverter til numerisk format.
    vindhastighed = as.numeric(vindhastighed),     # Konverter til numerisk format.
    lufttemperatur = as.numeric(lufttemperatur),   # Konverter til numerisk format.
    sigtbarhed = as.numeric(sigtbarhed),           # Konverter til numerisk format.
    skydække = as.numeric(skydække),               # Konverter til numerisk format.
    skyhøjde = as.numeric(skyhøjde),               # Konverter til numerisk format.
    nedbør = as.numeric(nedbør)                    # Konverter til numerisk format.
  )

# ------------------------------------------------------------------------
# Dataopsummering
# ------------------------------------------------------------------------
# Beregn og print en række nøgleoplysninger om datasættet for at sikre, at det er klar til

# Antal observationer (rækker)
antal_observationer <- nrow(joins_samlet_model)
print(paste("Antal observationer:", antal_observationer))
```

```
[1] "Antal observationer: 98"
```

```
# Antal numeriske variabler
antal_numeriske <- joins_samlet_model |>
  dplyr::select(where(is.numeric)) |>
  ncol()
print(paste("Antal numeriske variabler:", antal_numeriske))
```

```
[1] "Antal numeriske variabler: 14"
```

```
# Antal kategoriske variabler
antal_kategoriske <- joins_samlet_model |>
  dplyr::select(where(is.factor)) |>
  ncol()
print(paste("Antal kategoriske variabler:", antal_kategoriske))
```

```
[1] "Antal kategoriske variabler: 4"
```

```
# Andelen af manglende værdier (efter imputering)
missing_values <- colSums(is.na(joins_samlet_model))  # Beregn antal manglende værdier per
missing_percentage <- round((sum(missing_values) / (antal_observationer * ncol(joins_samle
print(paste("Andel af manglende værdier (efter imputering):", missing_percentage, "%"))
```

```
[1] "Andel af manglende værdier (efter imputering): 0 %"
```

```
#==============================================================================
#                                5. Modeller
#==============================================================================
# 0-Feature Model (Baseline)
```

```r
# En baseline-model uden nogen prædiktorer.
# Denne model forudsiger altid middelværdien af målvariablen (Guld_menu_stk).
# Bruges som referencepunkt for mere avancerede modeller.

set.seed(42)
train <- sample(1:nrow(joins_samlet_model), nrow(joins_samlet_model) * 2/3)  # 2/3 af data
test <- (-train)  # Resten som testdata.
y <- joins_samlet_model$guld_menu_stk  # Definer målvariabel.
y.test <- y[test]  # Definer målvariabel for testdata.
glm.fit <- glm(guld_menu_stk ~ 1, data = joins_samlet_model[train, ])  # Baseline-model.
rmse_0_cv <- sqrt(cv.glm(joins_samlet_model[train, ], glm.fit, K = 5)$delta[1])  # RMSE fr
rmse_0_test <- sqrt(mean((y.test - predict(glm.fit, joins_samlet_model[test, ]))^2))  # RM

rmse_0_test  # Udskriv RMSE for baseline-modellen.
```

```
[1] 147.4907
```

```r
# ------------------------------------------------------------------------
# Ridge Regression
# ------------------------------------------------------------------------
set.seed(42)

# Definer designmatrix og målvariabel for Ridge Regression.
x <- model.matrix(guld_menu_stk ~ ., data = joins_samlet_model)[, -1]  # Fjern intercept.
y <- joins_samlet_model$guld_menu_stk  # Målvariabel.
grid <- 10^seq(10, -2, length = 100)  # Generer 100 lambda-værdier mellem 10^10 og 10^-2.

# Ridge-regression med krydsvalidering.
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid)  # Ridge regression.
```

```r
cv.ridge <- cv.glmnet(x[train, ], y[train], alpha = 0, lambda = grid)  # Krydsvalidering.
bestlam_ridge <- cv.ridge$lambda.min  # Optimal lambda.

# Forudsigelser og beregning af RMSE for testdata.
ridge_pred <- predict(ridge.mod, s = bestlam_ridge, newx = x[test, ])  # Forudsigelser.
rmse_ridge_test <- sqrt(mean((ridge_pred - y.test)^2))  # RMSE for testdata.
cat("RMSE for Ridge Regression:", rmse_ridge_test, "\n")  # Udskriv RMSE.
```

```
RMSE for Ridge Regression: 125.6702
```

```r
# Udtræk koefficienter for den optimale lambda.
ridge_coefs <- coef(ridge.mod, s = bestlam_ridge)

# Konverter koefficienterne til en læsbar data frame.
ridge_coefs_df <- data.frame(
  Variable = rownames(ridge_coefs),           # Navne på variabler.
  Coefficient = as.numeric(ridge_coefs)      # Værdier af koefficienter.
)

# Fjern variabler med 0-koefficienter (for bedre overblik).
ridge_coefs_df <- ridge_coefs_df[ridge_coefs_df$Coefficient != 0, ]

# Print resultaterne.
print(ridge_coefs_df)
```

```
                    Variable   Coefficient
1                (Intercept)  5.367465e+02
2                ugedagMonday  1.208348e+01
```

```
3                  ugedagSaturday -3.487309e+01
4                    ugedagSunday  4.011740e+00
5                  ugedagThursday  3.117635e+01
7                    aarstidForår  1.970954e+01
8                   aarstidSommer -3.403219e+00
9                  aarstidEfterår  9.321045e+00
10       guld_menuer_sidste_møde  2.650919e-01
11            guld_menu_kategoriB -1.025812e+01
12            guld_menu_kategoriC -3.346501e+01
13 modstander_kategoriMidterklub -4.489984e+00
14   modstander_kategoriBundklub -6.684095e+00
15          point_sidste_3_kampe  6.233034e+00
16               antal_tilskuere  2.379381e-02
17         tilskuere_sidste_møde  8.106009e-03
18                   vindretning  1.358841e-01
19                      lufttryk -2.094732e-01
20                 luftfugtighed -5.609546e-01
21                 vindhastighed -1.420013e+00
22                lufttemperatur  3.433348e-01
23                    sigtbarhed  7.859874e-04
24                      skydække -1.667132e-01
25                      skyhøjde  6.813334e-03
26                        nedbør -2.324955e+00
```

```r
# ------------------------------------------------------------------------
# Lasso Regression
# ------------------------------------------------------------------------
set.seed(42)
```

```r
# Designmatrix og målvariabel for Lasso Regression.
x <- model.matrix(guld_menu_stk ~ ., joins_samlet_model)[, -1]  # Skaber designmatrix.
y <- joins_samlet_model$guld_menu_stk  # Målvariabel (genbrugt fra før).
grid <- 10^seq(10, -2, length = 100)  # Generer 100 lambda-værdier mellem 10^10 og 10^-2.

# Lasso-regression med krydsvalidering.
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = grid, thresh = 1e-12)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1, lambda = grid, nfolds = 10)
bestlam <- cv.out$lambda.min  # Optimal lambda.
rmse_lasso_cv <- sqrt(cv.out$cvm[cv.out$lambda == bestlam])  # RMSE fra krydsvalidering.
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test, ])  # Forudsigelser for testda
rmse_lasso_test <- sqrt(mean((lasso.pred - y.test)^2))  # RMSE for testdata.

cat("RMSE for Lasso Regression:", rmse_lasso_test, "\n")  # Udskriv RMSE.
```

```
RMSE for Lasso Regression: 133.1096
```

```r
# Udtræk koefficienter fra Lasso-modellen.
lasso_coefs <- as.matrix(coef(lasso.mod, s = bestlam))  # Konverter til matrix.

# Opret en data frame med ikke-nul koefficienter.
lasso_coefs_df <- data.frame(
  Variable = rownames(lasso_coefs),  # Hent variabelnavne.
  Coefficient = as.numeric(lasso_coefs)  # Hent koefficientværdier.
) |>
  filter(Coefficient != 0)  # Filtrer kun variabler med ikke-nul koefficienter.

# Print resultatet.
print(lasso_coefs_df)
```

```
                 Variable   Coefficient
1             (Intercept) 188.631344091
2 guld_menuer_sidste_møde   0.379027809
3         antal_tilskuere   0.041026155
4             vindretning   0.041716332
5           luftfugtighed  -0.258881565
6              sigtbarhed   0.001178694
7                skydække  -0.067160245
```

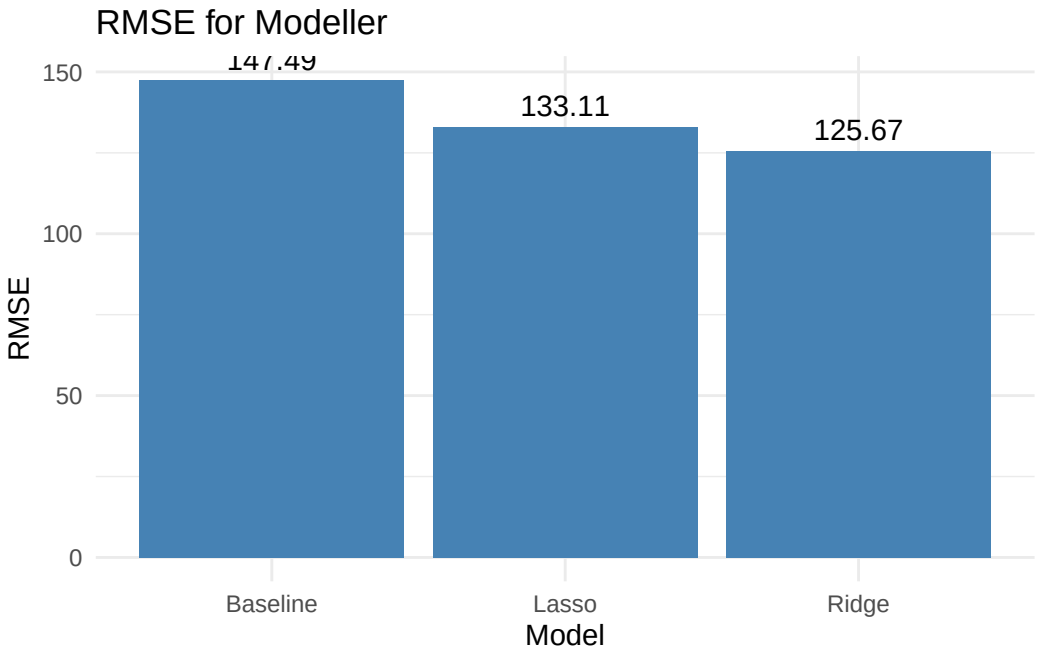
```r
# ------------------------------------------------------------------------
# Sammenligning af modeller
# ------------------------------------------------------------------------

# Beregn RMSE for hver model
baseline_rmse <- rmse_0_test  # RMSE for baseline-modellen.
ridge_rmse <- rmse_ridge_test  # RMSE for Ridge Regression.
lasso_rmse <- rmse_lasso_test  # RMSE for Lasso Regression.

# Beregn MAE (Mean Absolute Error) for hver model
baseline_mae <- mean(abs(y.test - predict(glm.fit, joins_samlet_model[test, ])))  # MAE fo
ridge_mae <- mean(abs(ridge_pred - y.test))  # MAE for Ridge Regression.
lasso_mae <- mean(abs(as.numeric(lasso.pred) - y.test))  # MAE for Lasso Regression.

# Beregn R² (forklaringsgrad) for hver model
baseline_r2 <- 1 - sum((y.test - predict(glm.fit, joins_samlet_model[test, ]))^2) / sum((y
ridge_r2 <- 1 - sum((y.test - ridge_pred)^2) / sum((y.test - mean(y.test))^2)  # R² for Ri
lasso_r2 <- 1 - sum((y.test - as.numeric(lasso.pred))^2) / sum((y.test - mean(y.test))^2)

# Opret en tibble til sammenligning af modeller
```

```r
results_table <- tibble(
  Model = c("Baseline", "Ridge", "Lasso"),  # Navne på modeller.
  RMSE = c(baseline_rmse, ridge_rmse, lasso_rmse),  # RMSE-værdier.
  MAE = c(baseline_mae, ridge_mae, lasso_mae),  # MAE-værdier.
  R_squared = c(baseline_r2, ridge_r2, lasso_r2)  # R²-værdier.
)

# Print resultattabel
print(results_table)
```

```
# A tibble: 3 x 4
  Model     RMSE   MAE R_squared
  <chr>    <dbl> <dbl>     <dbl>
1 Baseline  147. 120.   -0.00211
2 Ridge     126.  92.7   0.272
3 Lasso     133.  90.7   0.184
```

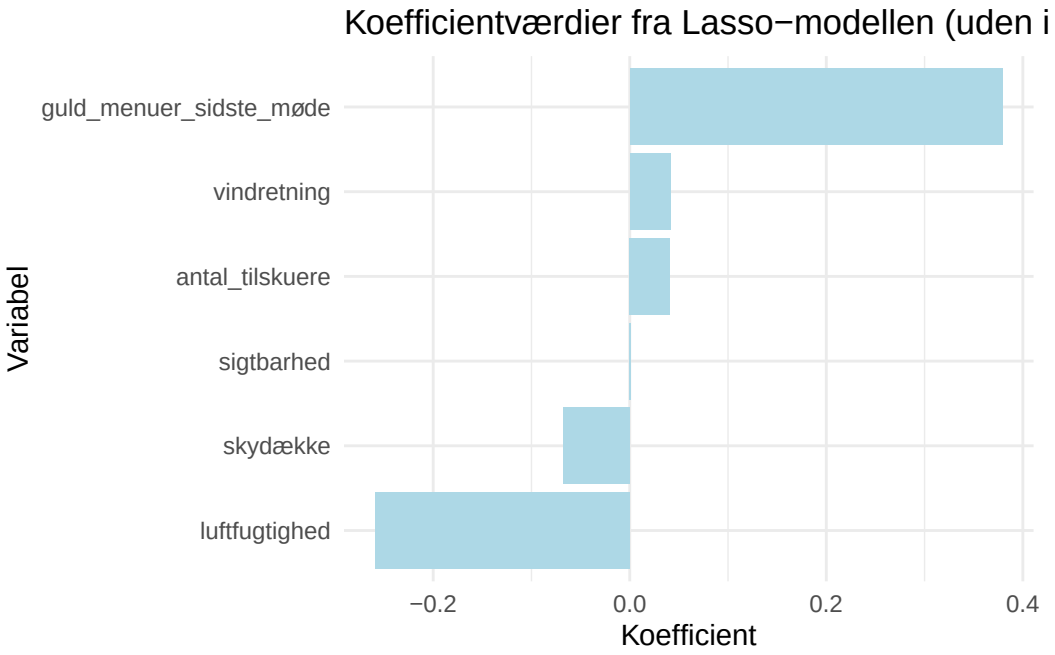
```r
# ------------------------------------------------------------------------
# Visualisering af RMSE
# ------------------------------------------------------------------------
# Diagram der sammenligner RMSE (Root Mean Square Error) for de forskellige modeller.
ggplot(results_table, aes(x = Model, y = RMSE)) +
  geom_col(fill = "steelblue") +  # Søjlediagram med farven 'steelblue'.
  geom_text(aes(label = round(RMSE, 2)), vjust = -0.5) +  # Tilføj RMSE-værdier som tekst
  labs(
    title = "RMSE for Modeller",  # Titel på plottet.
    x = "Model",                  # Navn på x-aksen.
    y = "RMSE"                    # Navn på y-aksen.
```

```
  ) +
  theme_minimal()  # Minimalistisk tema for et rent udseende.
```



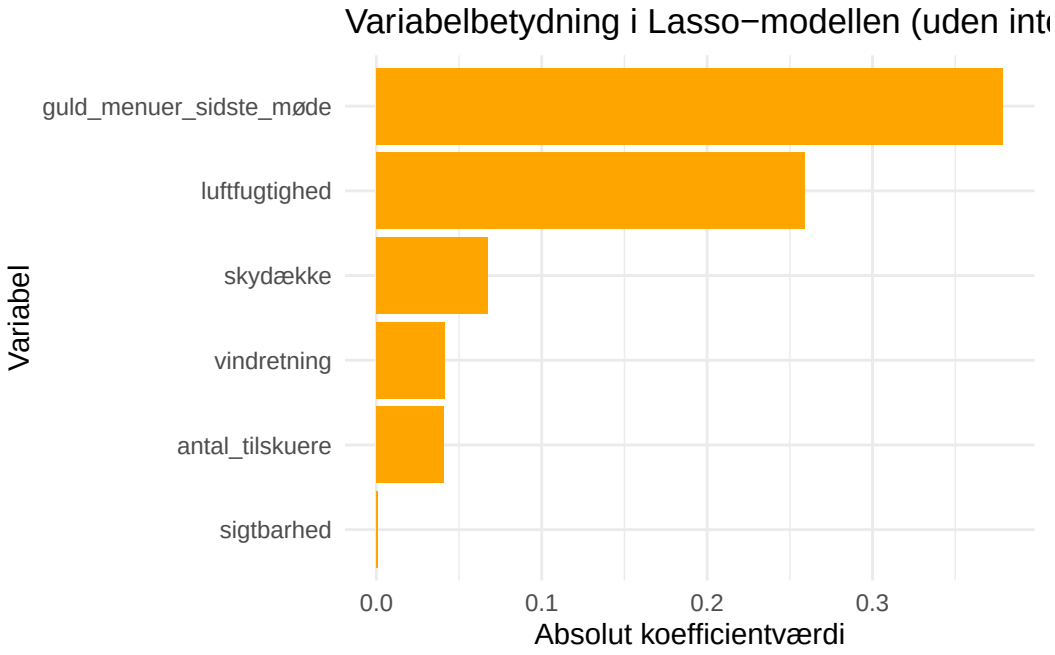

```
# ------------------------------------------------------------------------------
# Visualisering af Lasso-koefficienter
# ------------------------------------------------------------------------------

# Diagram, der viser værdierne af koefficienterne fra Lasso-modellen, eksklusiv intercept.
ggplot(lasso_coefs_df |> filter(Variable != "(Intercept)"),
       aes(x = reorder(Variable, Coefficient), y = Coefficient)) +
  geom_col(fill = "lightblue") +  # Søjlediagram med farven 'lightblue'.
  coord_flip() +  # Roter diagrammet horisontalt for bedre læsbarhed.
  labs(
    title = "Koefficientværdier fra Lasso-modellen (uden intercept)",  # Titel på plottet.
    x = "Variabel",                                                    # Navn på x-aksen.
    y = "Koefficient"                                                  # Navn på y-aksen.
```

```
  ) +
  theme_minimal()  # Minimalistisk tema for et rent udseende.
```




```
# ------------------------------------------------------------------------------
# Visualisering af Lasso-koefficienternes betydning
# ------------------------------------------------------------------------------

# Tilføj en kolonne med absolutværdien af koefficienterne for at vurdere betydningen.
lasso_coefs_df <- lasso_coefs_df |>
  mutate(Importance = abs(Coefficient))  # Beregn absolutværdien af koefficienterne.

# Diagram, der viser betydningen af variablerne baseret på absolutværdien af deres koeffic
ggplot(lasso_coefs_df |> filter(Variable != "(Intercept)"),
       aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_col(fill = "orange") +  # Søjlediagram med farven 'orange'.
  coord_flip() +  # Roter diagrammet horisontalt for bedre læsbarhed.
```

```
  labs(
    title = "Variabelbetydning i Lasso-modellen (uden intercept)",  # Titel på plottet.
    x = "Variabel",                                                 # Navn på x-aksen.
    y = "Absolut koefficientværdi"                                  # Navn på y-aksen.
  ) +
  theme_minimal()  # Minimalistisk tema for et rent udseende.
```




```
#=============================================================================
#                   6. Forudsigelser og evaluering
#=============================================================================


# Vi vælger at fortsætte med Lasso Regression, da den giver en god balance mellem
# modelpræcision og simplicitet. Selvom Ridge Regression opnår en lavere RMSE (126)
# sammenlignet med Lasso Regression (133), foretrækker vi Lasso Regression, fordi
# den reducerer kompleksiteten ved at inkludere kun 6 variabler i modellen.
# Dette gør Lasso Regression lettere at tolke og implementere.
```

```r
#------------------------------------------------------------------------------
#                          Lasso Regression Prediction
#------------------------------------------------------------------------------

# Funktion til forudsigelse med Lasso Regression
lasso_predict <- function(nye_data, lasso_coefs_df) {
  # Hent interceptet
  intercept <- lasso_coefs_df$Coefficient[lasso_coefs_df$Variable == "(Intercept)"]

  # Fjern interceptet fra lasso_coefs_df for at behandle de andre variabler separat
  koefs_df <- lasso_coefs_df[lasso_coefs_df$Variable != "(Intercept)", ]

  # Tjek for manglende variabler i nye_data
  manglende_variabler <- setdiff(koefs_df$Variable, names(nye_data))
  if (length(manglende_variabler) > 0) {
    stop(glue::glue("Følgende variabler mangler i nye_data: {paste(manglende_variabler, co
  }

  # Opret en navngivet vektor med koefficienter
  koefs <- setNames(koefs_df$Coefficient, koefs_df$Variable)

  # Konverter nye_data til en numerisk vektor, der matcher rækkefølgen af koefficienterne
  nye_data_vector <- unlist(nye_data)[names(koefs)]

  # Tjek for eventuelle NA-værdier
  if (any(is.na(nye_data_vector))) {
    stop("De nye data indeholder NA-værdier.")
  }
```

```r
  # Beregn forudsigelsen som en lineær kombination af input og koefficienter + intercept
  forudsigelse <- sum(nye_data_vector * koefs) + intercept

  return(forudsigelse)
}

# Vi henter fremtidige vejrdata herfra:
# https://open-meteo.com/en/docs/dmi-api#latitude=56.4532&longitude=9.402&hourly=relative_
# Data for visibility kan vi kun finde predicted tre dage frem i tiden
# Under kampen har vi skrevet hvilken dato vi har taget vejrdata fra, da kampdatoen ligger

# Kampen mod Silkeborg
# Vejrdata d. 4. januar kl. 12.00
silkeborg <- list(
  guld_menuer_sidste_møde = 698,  # Antal guldmenuer fra sidste møde
  luftfugtighed = 93,             # Luftfugtighed i % (målt 2 meter over terræn)
  skydække = 28,                  # Totalt skydække i %
  vindretning = 271,              # Vindretning i grader
  antal_tilskuere = 5728,         # Forventet antal tilskuere som de kan forudsige med den
  sigtbarhed = 50000              # Sigtbarhed angivet i meter
)

# Kampen mod Vejle
# Vejrdata d. 5. januar kl. 12.00
vejle <- list(
  guld_menuer_sidste_møde = 771,  # Antal guldmenuer fra sidste møde
  luftfugtighed = 94,             # Luftfugtighed i % (målt 2 meter over terræn)
  skydække = 100,                 # Totalt skydække i %
```

```r
  vindretning = 122,              # Vindretning i grader
  antal_tilskuere = 6059,         # Forventet antal tilskuere som de kan forudsige med den
  sigtbarhed = 5900               # Sigtbarhed angivet i meter
)

# Kampen mod FC København
# Vejrdata d. 6. januar kl. 12.00
kobenhavn <- list(
  guld_menuer_sidste_møde = 650,  # Antal guldmenuer fra sidste møde
  luftfugtighed = 97,             # Luftfugtighed i % (målt 2 meter over terræn)
  skydække = 100,                 # Totalt skydække i %
  vindretning = 193,              # Vindretning i grader
  antal_tilskuere = 8092,         # Forventet antal tilskuere som de kan forudsige med den
  sigtbarhed = 2220               # Sigtbarhed angivet i meter
)

# Vi laver forudsigelsen
forventet_guldmenuer_silkeborg <- round(lasso_predict(silkeborg, lasso_coefs_df = lasso_co
forventet_guldmenuer_vejle <- round(lasso_predict(vejle, lasso_coefs_df = lasso_coefs_df))
forventet_guldmenuer_kobenhavn <- round(lasso_predict(kobenhavn, lasso_coefs_df = lasso_co

# Print resultatet
cat("Forventet antal guldmenuer afhentet ved kampen imod Silkeborg IF:", forventet_guldmen
```

```
Forventet antal guldmenuer afhentet ved kampen imod Silkeborg IF: 732
```

```r
cat("Forventet antal guldmenuer afhentet ved kampen imod Vejle BK:", forventet_guldmenuer_
```

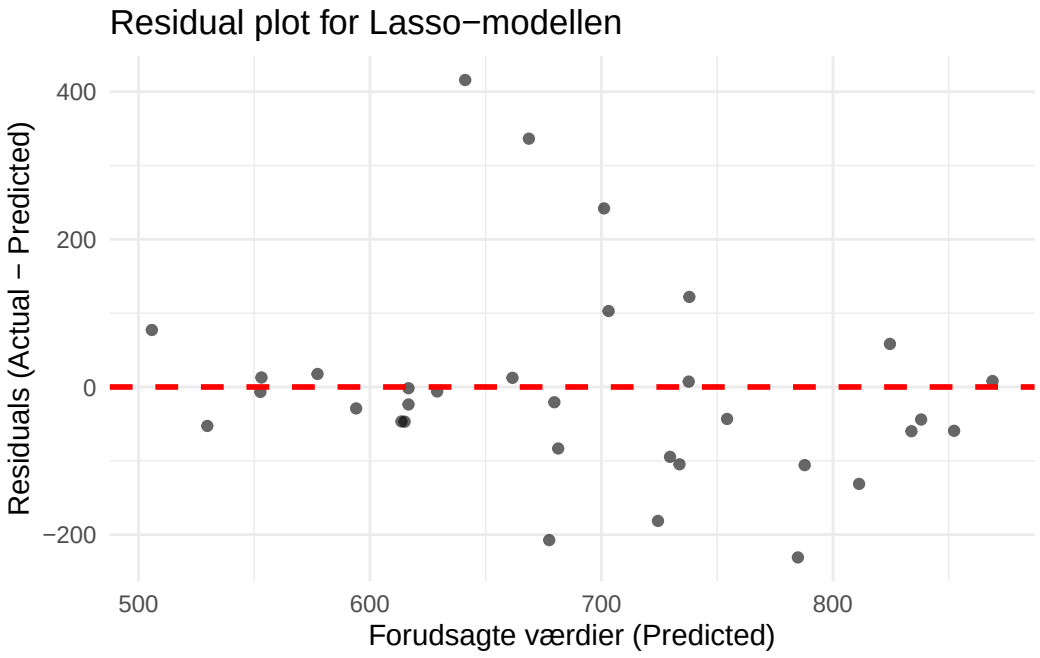
```
Forventet antal guldmenuer afhentet ved kampen imod Vejle BK: 710
```

```r
cat("Forventet antal guldmenuer afhentet ved kampen imod FC København:", forventet_guldmen
```

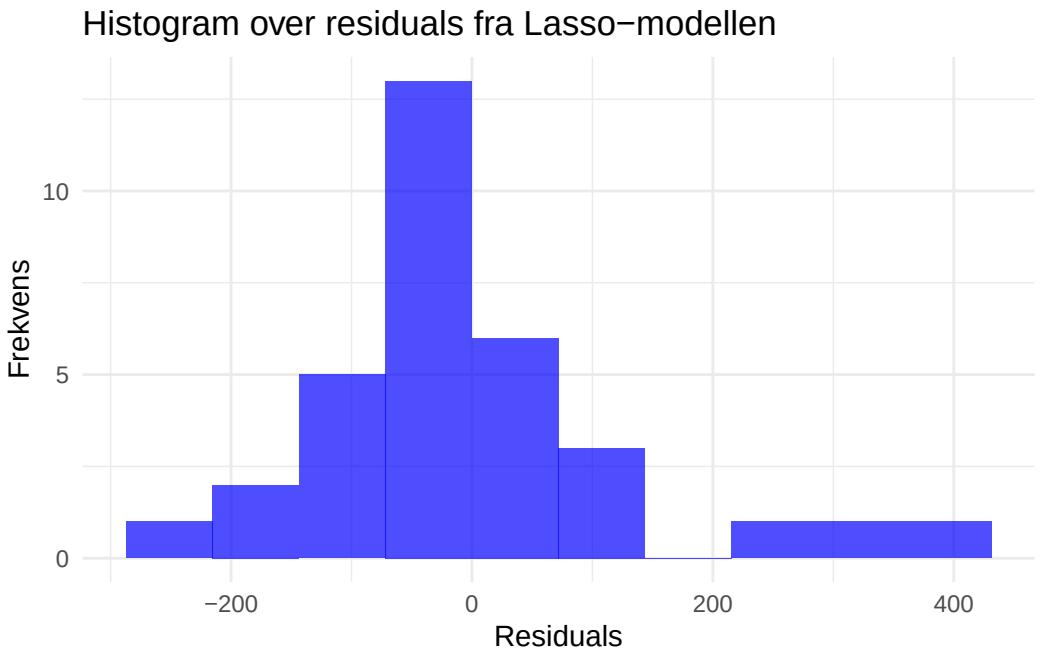
```
Forventet antal guldmenuer afhentet ved kampen imod FC København: 746
```

```r
# ------------------------------------------------------------------------------
# Residual plot for Lasso
# ------------------------------------------------------------------------------
# Residuals er forskellen mellem de faktiske værdier (y.test) og de forudsagte
# værdier fra Lasso-modellen (lasso.pred).
# De hjælper med at evaluere modellens præcision og afsløre potentielle problemer.
lasso_residuals <- y.test - as.numeric(lasso.pred)
# y.test er de faktiske værdier fra testdatasættet, mens as.numeric(lasso.pred)
# konverterer forudsagte værdier til numerisk format, hvis de ikke allerede er det.

# ------------------------------------------------------------------------------
# Opret en data frame til visualisering
# ------------------------------------------------------------------------------
# Vi opretter en data frame med tre kolonner:
# - `Actual`: De faktiske værdier (y.test).
# - `Predicted`: De forudsagte værdier fra modellen.
# - `Residuals`: Beregnede residuals (fejl).
lasso_residuals_df <- data.frame(
  Actual = y.test,                    # Faktiske værdier fra testdata.
  Predicted = as.numeric(lasso.pred),  # Forudsagte værdier fra modellen.
  Residuals = lasso_residuals          # Residuals, dvs. fejl (Actual - Predicted).
)

# ------------------------------------------------------------------------------
# Residual plot
```

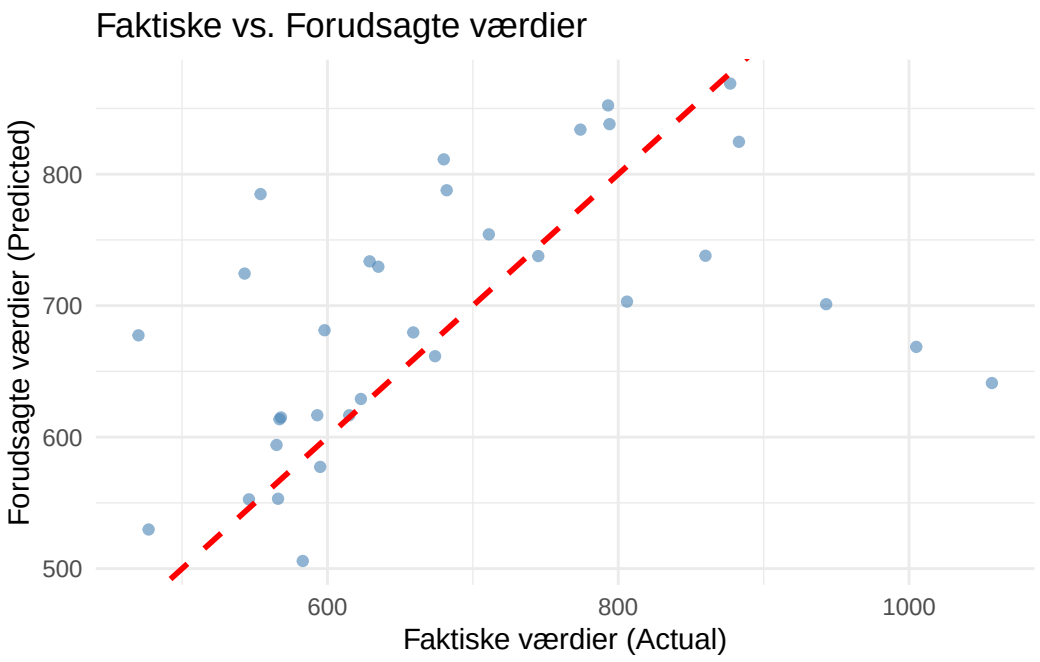
```r
# ------------------------------------------------------------------------------
# Visualisering af residualerne fra Lasso-modellen.
# Formålet med denne plot er at identificere, om der er systematiske fejl i modellen.

ggplot(lasso_residuals_df, aes(x = Predicted, y = Residuals)) +
  geom_point(alpha = 0.6) +  # Scatter plot med lidt transparens for bedre overblik.
  geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = 1) +
  labs(
    title = "Residual plot for Lasso-modellen",
    x = "Forudsagte værdier (Predicted)",
    y = "Residuals (Actual - Predicted)"
  ) +
  theme_minimal()
```



```r
# ------------------------------------------------------------------------------
# Histogram over residualer
# ------------------------------------------------------------------------------
# Histogrammet viser fordelingen af residualerne.
# En normalfordeling uden ekstremt skæve værdier indikerer en velfungerende model.

ggplot(lasso_residuals_df, aes(x = Residuals)) +
  geom_histogram(bins = 10, fill = "blue", alpha = 0.7, boundary = 0) +
  labs(
    title = "Histogram over residuals fra Lasso-modellen",
    x = "Residuals",
    y = "Frekvens"
  ) +
  theme_minimal()
```



```r
# ------------------------------------------------------------------------------
# Faktiske vs. forudsagte værdier
# ------------------------------------------------------------------------------
# Visualisering af sammenhængen mellem faktiske og forudsagte værdier.
# Ideelt bør punkterne ligge tæt omkring diagonallinjen y = x.

ggplot(lasso_residuals_df, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red", size = 1) +
  labs(
    title = "Faktiske vs. Forudsagte værdier",
    x = "Faktiske værdier (Actual)",
    y = "Forudsagte værdier (Predicted)"
  ) +
  theme_minimal()
```



```
# Evaluering af Lasso-modellen baseret på residual- og prædiktionsplot:
#
# Residualplot:
# - Residualerne er ikke tilfældigt fordelt omkring nul, hvilket tyder på, at modellen
#   ikke fanger al variation i dataene.
# - Der er tydelige outliers med høje residualer, både positive og negative, hvilket
#   indikerer, at nogle forudsigelser ligger langt fra de faktiske værdier.
# - Mønstre eller clustering i residualerne kan skyldes manglende forklarende variabler
#   i datasættet eller modelbegrænsninger.
#
# Histogram over residualer:
# - Residualerne er ikke normalfordelte, da fordelingen er let skæv.
# - Outliers, især i den positive retning, påvirker modellens præcision og robusthed.
# - En forbedring i modellens datagrundlag kan reducere skævheden og forbedre modellens yd
#
# Faktiske vs. forudsagte værdier:
# - Der er en vis afstand mellem faktiske og forudsagte værdier, hvilket indikerer, at mod
#   ikke præcist forudsiger for alle observationer.
# - Den diagonale linje viser, hvor præcise forudsigelserne skulle være, og mange punkter
#   afviger betydeligt fra denne linje.
# - Dette stemmer overens med modellens lave forklaringsgrad (18,4%) og høje RMSE (133,11)
#
# Overordnet anbefaling:
# - Modellen bør forbedres ved at inkludere flere relevante forklarende variabler, der kan
# - Overvej også alternative modeller eller kombinationer af modeller for at forbedre foru
```