UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ

DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

# PROIECT DE DIPLOMĂ

Laura Jianu

COORDONATOR ȘTIINȚIFIC

*Conf.univ.dr.ing. Cristian Mihăescu*

Iulie 2018

CRAIOVA

UNIVERSITATEA DIN CRAIOVA
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ

DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

# *APLICAȚIE PENTRU ANALIZA ACTIVITĂȚII UNEI COMPANII DE ÎNCHIRIERI BICICLETE FOLOSIND ”ENSEMBLE MODELS”*

Laura Jianu

COORDONATOR ȘTIINȚIFIC

*Conf.univ.dr.ing. Cristian Mihăescu*

*Iulie 2018*

CRAIOVA

*„Învățătura este o comoară care își urmează stăpânul pretutindeni.”*

Proverb popular

# DECLARAȚIE DE ORIGINALITATE

Subsemnatul *LAURA JIANU,* student la specializarea CALCULATOARE (CU PREDARE ÎN LIMBA ROMÂNĂ) din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

- cu titlul APLICAȚIE PENTRU ANALIZA ACTIVITĂȚII UNEI COMPANII DE ÎNCHIRIERI BICICLETE FOLOSIND "ENSEMBLE MODELS",
- coordonată de CONF.UNIV.DR.ING. CRISTIAN MIHĂESCU
- prezentată în sesiunea IULIE 2018.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
- însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
- precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului,

UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare şi Electronică

Departamentul de Calculatoare și Tehnologia Informației

Aprobat la data de
……
Șef de departament,
Prof. dr. ing.
Marius BREZOVAN

# PROIECTUL DE DIPLOMĂ

| | |
|---|---|
| Numele și prenumele studentului/-ei: | Jianu Laura |
| Enunțul temei: | *Aplicație pentru analiza activității unei companii de închirieri biciclete folosind "ensemble models" /Se dorește implementarea unei aplicații care să asigure analiza activității unei companii de închirieri biciclete în scop administrativ, din punct de vedere al estimării (predicției) numărului de închirieri.* |
| Datele de pornire: | *Realizarea unei aplicații WEB care să estimeze numărul de închirieri biciclete pentru compania în cauză folosind algoritmi din sfera "machine learning".* |
| Conținutul proiectului: | ***Enunțul temei*** *- specificarea cerintelor, a scopului proiectului și aplicabilitatea în viața de zi cu zi*<br>***Considerațiile teoretice*** *- prezentarea suportului teoretic utilizat in realizarea proiectului referitoare la tehnologii, medii de dezvoltare, etc.*<br>***Implementarea aplicației*** *- analiza datelor, codul sursă, organizarea claselor, interfața cu utilizatorul, etc.*<br>***Referințele*** *către materialele utilizate drept suport informativ*<br>***Concluzii*** |
| Material grafic obligatoriu: | *Prezentare PowerPoint*<br>*Diagrame UML*<br>*Capturi de ecran suport pentru funcționalitatea aplicației* |
| Consultații: | *Lunare* |
| Conducătorul științific (titlul, nume și prenume, semnătura): | Conf.univ.dr.ing. Cristian MIHĂESCU |
| Data eliberării temei: | 15.11.2017 |
| Termenul estimat de predare a proiectului: | 02.07.2018 |
| Data predării proiectului de către | 05.07.2018 |

| | |
|---|---|
| student și semnătura acestuia: | |

UNIVERSITATEA DIN CRAIOVA
Facultatea de Automatică, Calculatoare şi Electronică

Departamentul de Calculatoare și Tehnologia Informației

# REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

Numele și prenumele candidatului/-ei: *Jianu Laura*
Specializarea: *Calculatoare (cu predare în limba română)*
Titlul proiectului: *Aplicație pentru analiza activității unei companii de închirieri biciclete folosind "ensemble models"*

Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta):
- În facultate ☐
- În producție ☐
- În cercetare ☐
- Altă locație:

În urma analizei lucrării candidatului au fost constatate următoarele:

| | | | | | |
|---|---|---|---|---|---|
| Nivelul documentării | | Insuficient ☐ | Satisfăcător ☐ | Bine ☐ | Foarte bine ☐ |
| Tipul proiectului | | Cercetare ☐ | Proiectare ☐ | Realizare practică ☐ | Altul |
| Aparatul matematic utilizat | | Simplu ☐ | Mediu ☐ | Complex ☐ | Absent ☐ |
| Utilitate | | Contract de cercetare ☐ | Cercetare internă ☐ | Utilare ☐ | Altul |
| Redactarea lucrării | | Insuficient ☐ | Satisfăcător ☐ | Bine ☐ | Foarte bine ☐ |
| Partea grafică, desene | | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |
| Realizarea practică | Contribuția autorului | Insuficientă ☐ | Satisfăcătoare ☐ | Mare ☐ | Foarte mare ☐ |
| | Complexitatea Temei | Simplă ☐ | Medie ☐ | Mare ☐ | Complexă ☐ |
| | Analiza cerințelor | Insuficient ☐ | Satisfăcător ☐ | Bine ☐ | Foarte bine ☐ |
| | Arhitectura | Simplă ☐ | Medie ☐ | Mare ☐ | Complexă ☐ |
| | Întocmirea specificațiilor funcționale | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |

| | | | | | |
|---|---|---|---|---|---|
| | Implementarea | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |
| | Testarea | Insuficientă ☐ | Satisfăcătoare ☐ | Bună ☐ | Foarte bună ☐ |
| | Funcționarea | Da ☐ | Parțială ☐ | Nu ☐ | |
| Rezultate experimentale | | Experiment propriu ☐ | | Preluare din bibliografie ☐ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii și observații | | | | | |

În concluzie, se propune:

| ADMITEREA PROIECTULUI | RESPINGEREA PROIECTULUI |
|---|---|
| ☐ | ☐ |

Data, Semnătura conducătorului științific,

# REZUMATUL PROIECTULUI

Principalul obiectiv al lucrării este acela de a dezvolta o aplicație căreia atribuindu-i-se un set de date de antrenament, să poată realiza predicții pentru o companie de tip *bike-sharing*, metrici utile atât pentru proprietarul afacerii, cât și pentru cercetători, în studiul mobilității populației dintr-o anumită zonă geografică.

De asemenea, acest tip de predicție poate fi un ghid util pentru proprietarul unei astfel de companii în scopul de a selecta strategiile potrivite de marketing (urmărirea și exploatarea *trend*-urilor în funcție de sezon), cât și pentru eficientizarea resurselor necesare în cadrul planului de afaceri al companiei, prin intermediul aplicației estimându-se numărul de închirieri.

Modelul obținut prin predicție va reprezenta o sursă constantă de informare a proprietarului afacerii (al cărui acces la aplicație este exclusiv) asupra *trend*-urilor, dar și deciziilor pe care ar trebui să le ia pentru a evolua și obține profit, plasându-se cu un pas înaintea concurenței.

Predicția se va realiza după principiul *train-test*, adică, după antrenarea modelului cu datele furnizate de *Kaggle*, odată validat, acesta va putea realiza o predicție folosind algoritmul *Random Forest* din WEKA, predicție a cărei acuratețe va fi validată în mod oficial de cei de la *Kaggle* și ulterior introdusă într-un clasament universal.

Platforma *Kaggle* este exclusiv dedicată "*machine learning*", având propriile metrici de evaluare aplicată rezultatelor fiecărui rezolvitor în parte, în urma încărcării rezultatelor pe platformă (RMSLE, RMSE) etc. spre a realiza o clasificare coerentă, matematică.

Tehnologiile/limbajele de programare utilizate în cadrul proiectului sunt: *JAVA, JSP, HTML, CSS, BOOTSTRAP, WEKA, JAVASCRIPT*, iar ca și algoritmi de *Machine Learning* menționez *ZeroR, Decision Trees* și în principal *Random Forest*.

Pe parcursul realizării proiectului am acumulat experiență în proiectarea și dezvoltarea aplicațiilor web, am interacționat cu tehnologii și medii de dezvoltare noi și mi-am aprofundat cunoștințele în domeniul *Machine Learning*, în continuă expansiune în prezent. Problemele ivite pe parcurs au fost datorate interacționării cu noile tehnologii, respectiv a incompatibilităților între acestea și mediile de dezvoltare, dar cu răbdare și perseverență a apărut și soluționarea lor.

***Termenii cheie***: *MACHINE LEARNING, JAVA, WEKA, RANDOM FOREST, TRAIN DATA, TEST DATA, PREDICTION, ECLIPSE*.

# PROLOG

# CUPRINSUL

# LISTA FIGURILOR

# LISTA TABELELOR

# 1 INTRODUCERE

## 1.1 Scopul

Scopul acestui proiect este de a realiza o aplicație utilă pentru o companie de închirieri biciclete din punct de vedere administrativ, mai precis, estimarea numărului de închirieri pe baza unor tipare, *trend*-uri remarcate din perioade anterioare, factori ce vor influența în mod direct această metrică.

Predicția obținută cu ajutorul aplicației va constitui pentru proprietarul/administratorul afacerii un instrument esențial în eficientizarea resurselor (umane, materiale, financiare, etc.) deoarece îi va permite să fie pregătit în avans cu cererea pieței din viitorul apropiat. De exemplu, pe baza estimărilor, va putea ști când să își suplimenteze numărul de biciclete, astfel că nu va risca în ceea ce privește avantajul concurenței, respectiv "migrarea/exodul" clienților către rivali.

Aplicația poate fi chiar extinsă și utilizată și în alte domenii, prin ajustarea parametrilor și colectarea datelor necesare, păstrându-și însă utilitatea generală, aceea de a realiza predicții, viitor suport al strategiilor de *business*.

## 1.2 Motivația

Am ales acest subiect din dorința de a-mi îmbunătăți cunoștințele în ceea ce privește conceptul de *Machine Learning*, dat fiind că în ultima perioadă acesta a devenit un punct de atracție în "universul" Inteligenței Artificiale, considerat drept un element-cheie în desfășurarea diverselor activități cotidiene. De la servicii financiare, sănătate, activități de ordin guvernamental, transport, extracții de resurse minerale, turism, până la marketing, strategii de *business* ș.a., tot ceea ce implică un volum mare de date se îmbină armonios cu algoritmii care "construiesc" modele, descoperă relații între atribute, șabloane, tipare comportamentale, trenduri care susțin eficientizarea proceselor și facilitează luarea unor decizii eficace fără prea multă intervenție umană, dar bazate pe un fundament matematic solid.

Per total, *Machine Learning*-ul este un domeniu în perpetuă expansiune, ofertant și care constituie un subiect foarte interesant de abordat și în domeniul academic, realizând legătura între abstract, matematic și lumea reală.

# 2 NOȚIUNI TEORETICE ȘI TEHNOLOGII UTILIZATE

În cadrul proiectului am utilizat diverse concepte/noțiuni teoretice caracteristice domeniului de studiu, tehnologii, limbaje de programare, librării, medii de dezvoltare, etc., unele dintre acestea fiindu-mi străine până la dezvoltarea acestei aplicații.

Pentru o perspectivă mai clară asupra temei alese și a domeniului din care face parte, în continuare voi prezenta cele mai sus menționate.

## 2.1 Machine Learning

### 2.1.1 Generalități

*Machine Learning* sau Învățarea Automată este un domeniu al *computer science* care folosește, de regulă, tehnici statistice pentru a acorda calculatoarelor posibilitatea de "a învăța" cu date (set de valori ale unor variabile cantitative/calitative) fără a fi explicit programate să realizeze acest lucru. Acest domeniu se ocupă cu studiul și construcția algoritmilor care pot "învăța", "asimila cunoștiințe" pe baza unui set de date furnizat ca date de intrare, astfel realizând apoi o serie de predicții prin construcția unui model din mostrele de *input* (date de intrare). Un model (matematic) reprezintă o descriere a unui sistem folosind un limbaj și concepte de ordin matematic.

Denumirea i-a fost acordată în 1959 de Arthur Samuel, pionier de origine americană în domeniul *computer science* și al Inteligenței Artificiale. Programul dezvoltat de acesta, "Samuel Checkers-playing Program" a fost printre primele programe de tipul *self-learning* (autoînvățare) de succes, fiind considerat o demonstrație primară a conceptelor fundamentale de Inteligență Artificială.

*Machine Learning*-ul, ca și domeniu de cunoștiințe este înrudit cu statistica computațională (*computational statistics*) care se orientează, de asemenea, către realizarea predicțiilor cu ajutorul computerelor. În ceea ce privește analiza datelor (*data analytics), Machine Learning*-ul este o metodă utilizată pentru descompunerea/scindarea modelelor complexe și a algoritmilor care în sine conduc la predicții; în scopuri comerciale, această analiză împreună cu diverse tehnici de *Machine Learning* poartă denumirea generică de *predictive analysis* (analiză predictivă). Aceste modele analitice oferă posibilitatea cercetătorilor, inginerilor, analiștilor și celor din domeniul *data science* să "producă rezultate repetabile și de încredere", descoperind astfel "secrete" din diversele relații, trend-uri și tipare în cadrul setului de date.

Tom M. Mitchell a furnizat o definiție formală și unanim acceptată a algoritmilor studiați în domeniul *Machine Learning*: "Un program de calculator învață dintr-o experiență E ținând cont de o clasă de *task*-uri T și măsura de performanță P la realizarea *task*-urilor din clasa T dacă performanța la *task*-urile din T, măsurată prin P se îmbunătățește cu experiența E."

### 2.1.2 Task-uri de tip Machine Learning

*Task*-urile din sfera *Machine Learning* sunt în general clasificate în două mari categorii depinzând dacă există un "semnal" de învățare sau un "*feedback*" disponibil aparținând sistemului de învățare:

- **Învățare supervizată (*Supervised Learning*):** Computerului i se prezintă exemple de *input* și modelele de *output* dorit de către "o entitate avizată" , iar scopul este să învețe o regulă generală ce realizează o legătură/dependență/funcție matematică intre input, respectiv output.
  - **Învățare semi-supervizată (*Semi-supervised learning*):** Computerului i se oferă un set incomplete de date de antrenament.
  - **Învățare activă (*Active learning*)**: Computerul poate obține doar etichete pentru un set limitat de instanțe, fiind necesară, de asemenea, optimizarea alegerilor în legătură cu obiectele selecționate.
  - **Învățare de tip consolidare (*Reinforcement learning*)**: Datele de antrenament (sub forma de recompensă, respectiv pedeapsă) sunt oferite doar ca *feedback* al acțiunilor programului într-un mediu dinamic. (ex: la conducerea unui autovehicul, jocuri, etc.)
- **Învățare nesupervizată** (***Unsupervised learning***): În acest caz, algoritmului de clasificare nu îi este acordată nicio etichetă (*label*) cu privire la datele de antrenament, lăsând pe seama acestuia acest *task*. (astfel se descoperă diverse șabloane/tipare în datele de *input*)

### 2.1.3 Aplicații de tip Machine Learning

Un alt tip de clasificare a *task*-urilor de tip *Machine Learning* apare datorită tipului de *output* ce se dorește a fi obținut de la un sistem de Învățare Automată:

- În **clasificare (*classification*),** *input*-urile sunt împărțite în două sau mai multe clase, iar sistemul care "învață" trebuie să producă un model ce asociază aceste date de *input* uneia sau mai multor clase. (*multi-label classification/supervised problem*)

- În **regresie (*regression*)**, *output*-urile sunt majoritar continue (valorile posibile sunt descrise folosind intervale pe axa numerelor reale, valorile nu pot fi numărate) și nu discrete (valori numărabile).
- În ***clustering,*** un set de *input*-uri (obiecte) se împarte în grupuri (*clusters*) astfel încât obiectele din același grup sunt similare unele cu celelalte și nu cu obiectele din alt *cluster* (clustere necunoscute în prealabil - *unsupervised problem*).
- **Estimarea densității** (***density estimation***) se ocupă de calcularea distribuției *input*-urilor într-un spațiu (funcție matematică ce descrie probabilitățile de apariție în cadrul unui eveniment).
- ***Dimensionality reduction*** simplifică *input*-ul, mapându-l într-un spațiu multi-dimensional mai mic. (ex: date din *input* care acoperă aceeași arie de interes).

Pentru un sistem/algoritm menit să "acumuleze cunoștiințe" (algoritm de învățare), "a învăța cum să învețe" (*learning to learn*/*meta-learning*) depinde de propriul "*inductive bias*", bazat pe experiența anterioară, deja acumulată, un fel de *curriculum vitae* (set de presupuneri pentru predicții).

### 2.1.4 Este *Machine Learning*-ul domeniul momentului ?

Datorită progresului tehnologic în domeniul computerelor, *Machine Learning*-ul contemporan prezintă noi valențe, comparativ cu forma sa din "punctul de pornire". Originea acestui domeniu este ceea ce numim "*pattern recognition*" (recunoașterea șabloanelor), respectiv teoria conform căreia computerele pot învăța să execute diverse sarcini fără a fi explicit programate să facă acest lucru. Teoria a fost dezvoltată de cercetătorii în domeniul Inteligenței Artificiale, care, după numeroase experimente, au dorit să afle dacă un calculator poate "învăța" dintr-o colecție de date.

Aspectul iterativ al acestui domeniu de studiu este unul foarte important deoarece algoritmul de învățare este mereu expus unui nou set de date de intrare și trebuie să fie capabil să se auto-adapteze. Bazat pe experiența acumulată în urma interacțiunii anterioare cu seturi de date, algoritmul de învățare al unui sistem produce rezultate repetabile și coerente simultan cu perfecționarea caracteristicii "*learning how to learn*" în vederea optimizării resurselor și a acurateții.

Majoritatea algoritmilor *Machine Learning* sunt utilizați de foarte mult timp, însă ceea ce este inedit și în continuă expansiune în acest domeniu este abilitatea de a efectua în mod automat calculele matematice complexe în mod continuu, din ce în ce mai rapid, utilizând un volum mare de date (*big data*).

Tehnologiile de ultimă generație, considerate chiar aplicațiile momentului în domeniul *high-tech* se bazează, nu în mod surprinzător, pe algoritmi și principii de *Machine Learning*.

Printre acestea se numără:

- Mașina autonomă Google, în prezent Waymo (*new way forward in mobility*) - *Machine Learning* în esență

- Sistemele de recomandări online/Marketing personalizat - Companii precum Amazon folosesc algoritmi de învățare pentru consolidarea poziției pe piață în ceea ce privește noțiunea de *client-oriented,* urmărind astfel păstrarea statului de client al său.

- Securitatea datelor - Kasperski, Deep Instinct, etc. folosesc în laboratoarele lor algoritmi de tip *Machine Learning* ce identifică similitudinile din codul sursă al programelor malware sau identifica șabloane în modul de accesare a datelor din *Cloud*.

- Tranzacții financiare - estimări ale stocurilor, acțiuni la bursă, etc.

- Căutările online - Google și competitorii săi studiază comportamentul utilizatorilor la interacțiunea cu motorul de căutare pentru a evalua relevanța rezultatelor afișate (click pe primul link sau navigare pe următoarele). Din experiența mai multor useri, motorul "învață" ce să corecteze pentru viitor.

ș.a.

### 2.1.5 Inteligență Artificială vs. *Machine Learning* sau Inteligență Artificială <u>SI</u> *Machine Learning* ?

*Machine Learning*-ul și Inteligența Artificială sunt două noțiuni considerate fundamentul tehnologiei de ultimă oră. De aceea, de multe ori, în diverse contexte sunt utilizate în mod interschimbabil, în ideea că ar fi același lucru și deci s-ar putea substitui unul cu celălalt. În realitate, cele două domenii

sunt înrudite (ambele se pretează la discuție când apar subiecte de genul: *big data, data analysis, autonomous cars*, etc.), dar totuși distincte.

Pe scurt, cele două domenii ar putea fi definite astfel:

**Inteligența Artificială** este conceptul amplu, vast că "mașinăriile pot executa sarcini" pe care noi, ca ființe umane le considerăm "inteligente" în timp ce ***Machine Learning***-ul este o aplicare a principiilor Inteligenței Artificiale bazată pe ideea de a acorda "mașinăriilor" acces la date cu scopul de a învăța în mod autonom.

**Inteligența Artificială** este un concept prezent în istoria umanității din cele mai vechi timpuri – mituri grecești prezintă "oameni mecanici" special concepuți pentru a imita comportamentul uman. Primele computere proiectate în Europa au fost proiectate drept "mașinării logice" capabile să înlocuiască subiecți umani la sarcini precum calcule aritmetice și memorarea rezultatelor. De aici, inginerilor le-a surâs ideea de a construi un "creier mecanic". Odată cu progresul tehnologiei, dar, mai ales cu avansarea cunoștințelor omenirii în ceea ce privește modul cum funcționează mintea umană, conceptul de IA și-a schimbat oarecum forma. În loc să se concentreze pe realizarea unor calcule din ce în ce mai complexe, activitatea în acest domeniu s-a axat pe "umanizarea" modului de executare a sarcinilor, pe imitarea comportamentului uman. (o anumită naturalețe în acțiuni)

Două descoperiri importante au condus la apariția *Machine Learning*-ului datorită faptului că progresul în Inteligența Artificială se realizează cu o viteză uimitoare, acesta fiind considerat "domeniul momentului", **"The one who will change our world"**.

**Una dintre acestea** se datorează lui Arthur Samuel, care a realizat că ar fi posibil să antrenăm computerele să "învețe singure", alternativă la a le "învăța" totul despre lume și cum să abordeze diverse *task*-uri. **A doua**, mai recentă decât prima, a fost apariția Internetului și amplificarea volumului de informație digitală ce poate fi generată, stocată și analizată. Odată cu aceste inovații în tehnologie, cea mai eficientă cale pentru ingineri este programarea "mașinăriilor" să "gândească" precum subiecți umani și apoi să le conecteze la Internet pentru a le da acces la toata informația pe care să o "asimileze singure".

În concluzie, IA și ML sunt interdependente în ceea ce numim "tehnologia momentului", fiecare cu beneficiile și aportul propriu la evoluție și îmbunătățire a calității proceselor în diverse domenii.

### 2.1.6 Aplicațiile momentului care utilizează ML

1. **Sistemele de recunoaștere vocală SIRI & CORTANA -** Cu cât progresează, vor "învăța să înțeleagă/distingă" nuanțe/tonuri ale vocii**.**
2. **Facebook -** Algoritmii folosiți de rețeaua socială permit identificarea facială a figurilor familiare din lista de contacte, fără a mai fi nevoie ca tu să bifezi un *tag*.
3. **Google Maps -** analizează viteza mobilelor din trafic prin locații anonime preluate de pe telefoane mobile. Astfel, reduce timpul de călătorie, sugerând rute mai rapide.
4. **Google Search -** Cel mai important motor de căutare oferă sugestii și recomandări pe baza experienței acumulate din căutari anterioare. Google a introdus algoritmul ***Knowledge Graph*** utilizat pentru a descifra conținutul semantic al unei căutări.
5. **Gmail -** funcția de *smart reply*
6. **PayPal -** Singurul serviciu de plăți online care utilizează algoritmi de ML pentru a combate frauda. Analizează datele clienților și evaluează riscul**.**
7. **Netflix -** Întregul sistem de recomandare al companiei se bazează pe ML
8. **Uber -** ML este crucial pentru gigantul tech. Acesta folosește algoritmii de ML pentru a determina ora de sosire, locațiile de start cursă, etc.
9. **Lyst -** Cumpărături online în domeniul modei, unde, pentru a potrivi căutările clienților cu recomandări relevante, algoritmii de ML realizează comparații vizuale folosind *meta-tags*
10. **Spotify -** După ascultarea unei melodii apar ca numeroase recomandări melodii din aceeași gamă. ML se folosește *like*-uri și *dislike*-uri pentru "a prezice viitorul".

## 2.2 Regresia liniară

### 2.2.1 Generalități

"În problemele de **regresie** se dorește predicția unor valori continue. În statistică, **regresia liniară** este o abordare liniară în modelarea relației dintre o variabilă dependentă Y și una sau mai multe variabile independente X. În cazul în care există o singură variabilă independentă, regresia poartă numele de **regresie liniară simplă**, iar pentru variabile independente multiple, regresia devinde **regresie liniară multiplă**. De reținut că termenul se distinge de **regresia liniară multivariată**, unde are loc predicția unor variabile dependente multiple, nu a uneia singure.

În **regresia liniară**, relațiile sunt modelate folosind funcții de predicție liniară ale căror parametri de model necunoscuți sunt estimati din date. Astfel de modele sunt numite **modele liniare**. Cel mai frecvent, se presupune că media condiționată a răspunsului dat valorii variabilelor explicative (sau

predictorilor) este o funcție afină a acestor valori; mai puțin frecvent, se folosește mediana condiționată.

Ca toate formele de analiză de regresie, **regresia liniară** se concentrează pe distribuția probabilității condiționate a răspunsului luând în considerare valorile predictorilor, mai degrabă decât distribuția probabilității comune tuturor acestor variabile, care este domeniul analizei multivariate.

**Regresia liniară** a fost primul tip de analiză de regresie care a fost studiată riguros și a fost utilizată extensiv în aplicații practice. Acest lucru se datorează faptului că modelele care depind liniar de parametrii lor necunoscuți sunt mai ușor de potrivit decât modelele care sunt legate neliniar de parametrii lor și deoarece proprietățile statistice ale estimatorilor care rezultă sunt mai ușor de determinat.

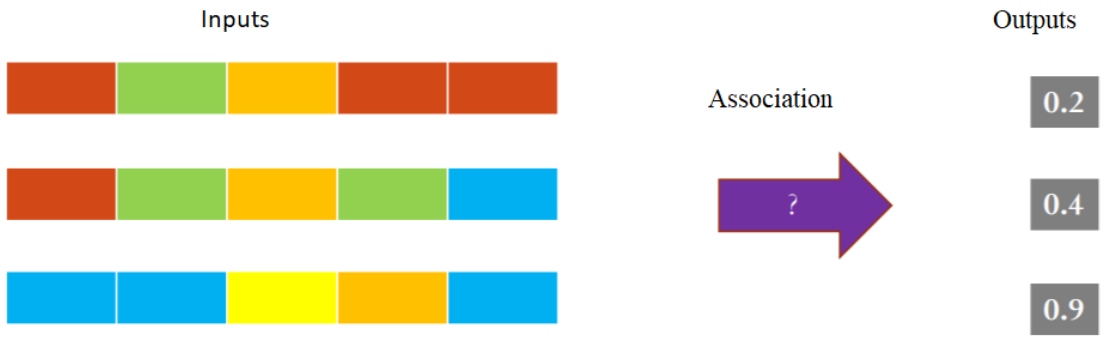


**Figura 1 – Regresia liniară**

**Regresia liniară** a fost primul tip de analiză de regresie care a fost studiată riguros și a fost utilizată extensiv în aplicații practice. Acest lucru se datorează faptului că modelele care depind liniar de parametrii lor necunoscuți sunt mai ușor de potrivit decât modelele care sunt legate neliniar de parametrii lor deoarece proprietățile statistice ale estimatorilor care rezultă sunt mai ușor de determinat.

**Regresia liniară multiplă:**

- m date de antrenament de tipul (X, Y):
  - X = (X^1 , X^2 , …, X^n ) un **vector de intrare**
  - n atribute - variabile independente, explicative, predictive

- Y – **ieșirea** (outcome)

- Variabila dependentă, funcție de celelalte variabile, variabila răspuns
- Y ia valori continue

- **Ecuația de regresie**: f(X) = b0 + b1 X^1+b2 X^2 + … + bn X^n
  - b1, b2, …, bn – coeficienții (parametrii) de regresie
  - b0 – interceptorul, constanta de regresie

**Regresia liniară** are multe utilizări practice. Cele mai multe aplicații se încadrează în una din următoarele două mari categorii:

- Dacă scopul este predicția, prognoza sau reducerea erorilor, **regresia liniară** poate fi folosită pentru a se potrivi unui model predictiv cu un set de date observat de valori ale răspunsului. După elaborarea unui astfel de model, dacă se colectează valori suplimentare ale variabilelor explicative fără o valoare de răspuns însoțitoare, modelul montat poate fi folosit pentru a face o predicție a răspunsului.

- Dacă scopul este de a explica variația variabilei de răspuns care poate fi atribuită variației variabilelor explicative, analiza de **regresie liniară** poate fi aplicată pentru a cuantifica puterea relației dintre răspuns și variabilele explicative și, în special, pentru a determina dacă unele variabilele explicative nu pot avea nicio relație liniară cu răspunsul sau pentru a identifica care subseturi de variabile explicative pot conține informații redundante despre răspuns.

**O linie de tendință** reprezintă o tendință, mișcarea pe termen lung a datelor din serii de timp după ce au fost luate în considerare alte componente. Ea arată dacă un anumit set de date (de exemplu, PIB, prețul petrolului sau prețul acțiunilor) a crescut sau a scăzut în perioada de timp. O linie de tendință ar putea fi trasată pur și simplu printr-un set de puncte de date, dar mai bine poziția și panta lor este calculată folosind tehnici statistice, cum ar fi **regresia liniară**. Liniile de trend sunt în mod obișnuit linii drepte, deși unele variații folosesc polinoame înalte în funcție de gradul de curbură dorit.

**Liniile de trend** sunt folosite uneori în analizele de afaceri pentru a afișa modificări ale datelor în timp. Acest lucru are avantajul de a fi simplu. Liniile de tendință sunt adesea folosite pentru a susține că o anumită acțiune sau eveniment (cum ar fi antrenamentul sau o campanie publicitară) a provocat schimbări observate la un moment dat. Aceasta este o tehnică simplă și nu necesită un grup de control,

un design experimental sau o tehnică de analiză sofisticată. Cu toate acestea, ea suferă din cauza lipsei valabilității științifice în cazurile în care alte modificări potențiale pot afecta datele.

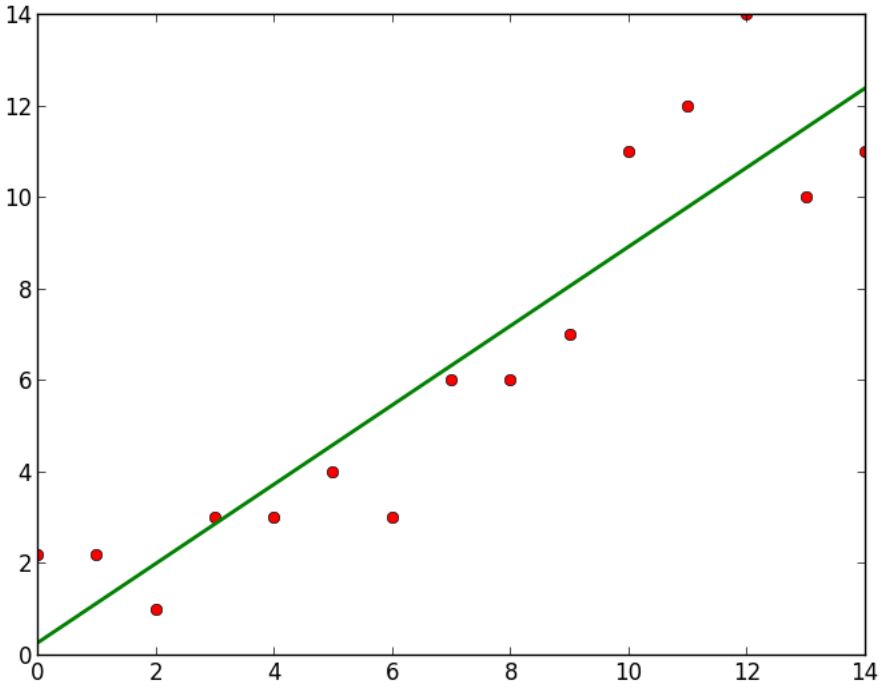


**Figura 2 - Exemplu regresie liniară**

### 2.2.2 Modele de regresie liniară

Modelele de **regresie liniară** sunt adesea “potrivite” (*fitted*) folosind abordarea **celor mai mici pătrate**, dar pot fi de asemenea potrivite și în alte moduri, cum ar fi minimizarea "lipsei de potrivire" în altă normă (ca și regresia cu abateri absolute minime) sau prin minimizarea unei versiuni “penalizate” a funcției de cost a celui mai mic pătrat ca în regresia creșterii. În schimb, abordarea celor mai mici pătrate poate fi utilizată pentru a se potrivi modelelor care nu sunt modele liniare. Astfel, deși termenii "cele mai mici pătrate" și "modelul liniar" sunt strâns legate, ele nu sunt sinonime.

**Un model de regresie liniară** “potrivit” (*fitted*) poate fi folosit pentru a identifica relația dintre o variabilă predictivă unică xj și variabila de răspuns y atunci când toate celelalte variabile ale predictorului din model sunt "ținute fixe". În mod specific, interpretarea lui βj este schimbarea

așteptată în y pentru o schimbare cu o singură unitate în xj atunci când celelalte covariate sunt menținute fixe - adică valoarea așteptată a derivatului parțial al y în raport cu xj. Acest lucru este uneori numit efectul unic al xj pe y. În contrast, efectul marginal al xj asupra y poate fi evaluat utilizând un coeficient de corelație sau un model simplu de **regresie liniară** care se referă numai la xj în raport cu y; acest efect este derivatul total al lui y în raport cu xj.

Întotdeauna trebuie să se țină seama de interpretarea rezultatelor **regresiei**, deoarece unii regresori nu pot permite modificări marginale (cum ar fi termenul de interceptare), în timp ce altele nu pot fi "ținute fixe"."

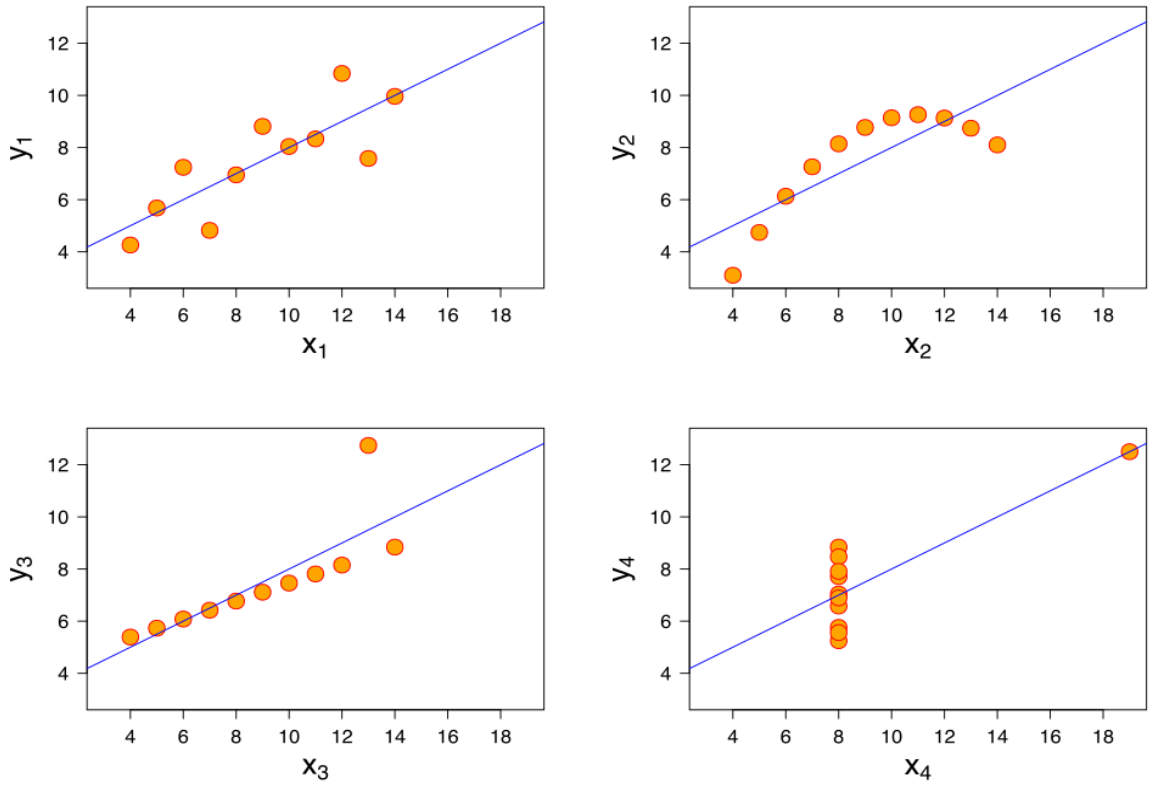


**Figura 3 - Seturile de date din cvartetul lui Anscombe sunt proiectate să aibă aproximativ aceeași dreaptă de regresie liniară (precum și mijloace aproape identice, deviații standard și corelații), dar sunt foarte diferite din punct de vedere grafic.**

### 2.2.3 RMSE/RMSLE

“**RMSE/RMSD (*Root-mean-square-error / Root-mean-square-deviation*)** Eroarea medie pătrată sau deviația media-pătrată a rădăcinii este o măsură frecvent utilizată a diferențelor dintre valori (eșantion și populație) sau un estimator și valorile efectiv observate. RMSD reprezintă deviația standard a eșantionului dintre diferențele dintre valorile estimate și valorile observate. Aceste diferențe individuale sunt numite reziduuri atunci când calculele sunt efectuate pe proba de date care a fost utilizată pentru estimare și se numesc erori de predicție atunci când sunt calculate în afara eșantionului.

RMSD servește pentru a agrega magnitudinea erorilor în predicții pentru diferite momente într-o singură măsură a puterii predictive. RMSD este o măsură de precizie, pentru a compara erorile de prognoză ale diferitelor modele pentru un anumit set de date și nu între seturi de date, deoarece este dependentă de scală.”

În cazul aplicației, evaluarea va fi realizată de cei de la **Kaggle** folosind varianta logaritmică a **RMSE** și anume **RMSLE**:

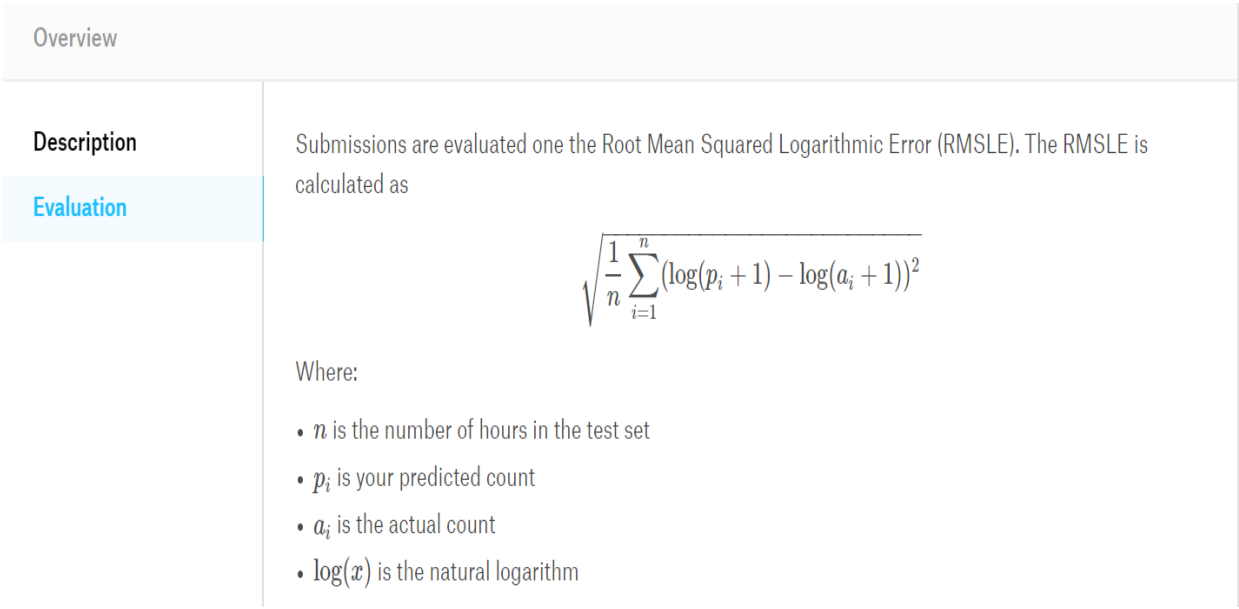


**Figura 4 - Metoda oficială de evaluare a Kaggle**

## 2.3 Ensemble learning / ensemble models

### 2.3.1 Generalități

Când doriți să achiziționați un produs, un telefon mobil de exemplu, veți merge până la primul magazin care îl comercializează și îl veți cumpăra pe baza sfaturilor vânzătorului ? Este foarte puțin probabil.

Cu siguranță urmăriți însă câteva *site*-uri unde oamenii au postat recenziile și au comparat diferite modele de telefoane, verificând caracteristicile și prețurile acestora. Vă veți consulta probabil și cu prietenii. Pe scurt, nu veți ajunge direct la o concluzie, ci veți lua o decizie luând în considerare și opiniile altor persoane.

**Ensemble models** în *Machine Learning* funcționează pe o idee similară. Acestea combină deciziile de la mai multe modele pentru a îmbunătăți performanța generală. Acest lucru se poate realiza în diferite moduri.

Din diverse contexte din viața de zi cu zi, deducem că un grup eterogen de persoane va lua decizii mai bune, mai eficiente comparativ cu indivizii. Realizând o paralelă la *Machine Learning*, teoria este valabilă și pentru un set divers de modele, spre deosebire de modelele unice. Această diversificare se realizează prin tehnica numită **ensemble learning**.

"În statistică și *Machine Learning*, **ensemble methods** utilizează algoritmi de învățare multipli pentru a obține o performanță predictivă mai bună, care ar putea fi obținută numai de la oricare dintre algoritmii de învățare constituenți.

Empiric, ansamblurile au tendința de a obține rezultate mai bune atunci când există o diversitate semnificativă între modele. Prin urmare, multe din **ensemble methods** încearcă să promoveze diversitatea dintre modelele pe care le combină. Deși probabil non-intuitivi, algoritmii aleatorii (cum ar fi ***random forests***) pot fi folosiți pentru a produce un ansamblu puternic. Folosind o varietate de algoritmi puternici de învățare, s-a demonstrat că este mai eficientă această cale decât utilizarea unor tehnici care încearcă să modifice modelele pentru a promova diversitatea."

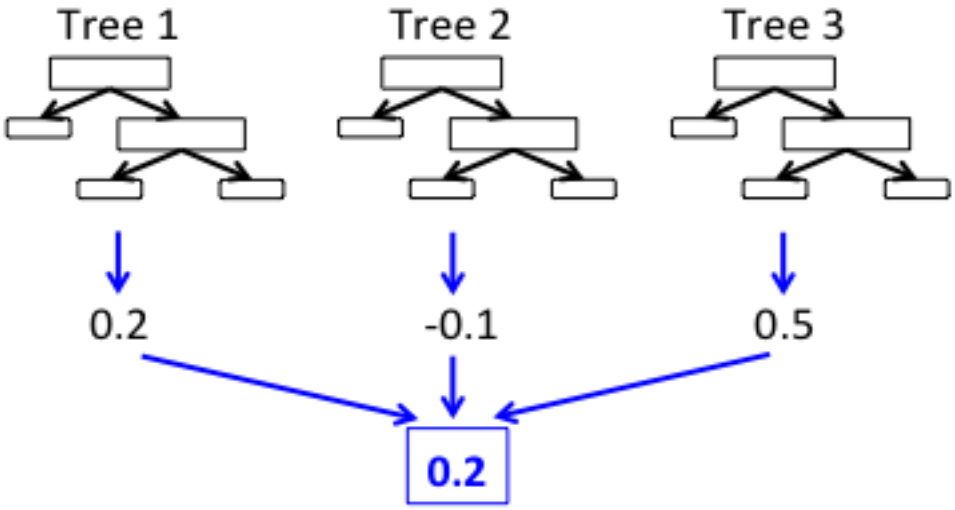


**Figura 5 - Exemplu de "vot" în *ensemble learning* (arbori)**

### 2.3.2 Tehnici de *ensemble learning*

- **Max voting**

**Max voting**-ul este folosit în general pentru problemele de clasificare. În cadrul acestei tehnici sunt folosite mai multe modele pentru a face predicții. Predicțiile fiecărui model sunt considerate drept "vot". Predicțiile pe care le primim de la majoritatea modelelor sunt folosite ca predicție finală.

De exemplu, dacă un grup de 7 persoane a acordat un vot distribuit astfel: 5 persoane au acordat 3 stele, iar 2 persoane 4 stele, majoritatea a acordat deci un rating de 3, deci evaluarea finală va fi luată ca având rezultatul 3.

- **Averaging**

Similară cu tehnica de max voting, la tehnica **averaging** se fac predicții multiple pentru fiecare punct de date în medie. La această metodă, se ia o medie a predicțiilor din toate modelele și se utilizează apoi pentru a face predicția finală. Valorile medii pot fi folosite pentru a face previziuni în probleme de regresie sau la calcularea probabilităților pentru probleme de clasificare. De exemplu, în cazul de mai sus, metoda de mediere ar lua media tuturor valorilor,

adică (3 + 3 + 3 + 3 + 3 + 4 + 4) / 7 = 3.28 .

- **Weighted average**

Aceasta este o extensie a metodei anterioare. Tuturor modelelor le sunt atribuite ponderi diferite care definesc importanța fiecărui model pentru predicție. De exemplu, dacă doi dintre subiecții ce acordă voturile sunt critici, în timp ce alții nu au o experiență anterioară în acest domeniu, atunci răspunsurile acestor doi au o importanță mai mare în comparație cu ceilalți.

Rezultatul este calculat astfel: [(3 * 0.2) + (3 * 0.2) + (3 * 0.2) + (3 * 0.2) + (3 * 0.2) + (4 * 0.3) + (4 * 0.3)] =... .

- **Bagging**

Ideea din spatele acestei tehnici este combinarea rezultatelor mai multor modele (de exemplu, toți arborii de decizie) pentru a obține un rezultat generalizat. Există însă șanse mari ca aceste modele să dea același rezultat, deoarece primesc aceeași intrare. Una dintre tehnicile utilizate pentru a rezolva această problemă este *bootstrapping*.

**Bootstrapping** este o tehnică de eșantionare în care se creează subseturi de observații din setul de date originale, cu înlocuire. Dimensiunea subseturilor este aceeași cu dimensiunea setului original.

Tehnica **Bagging** (sau **Bootstrap Aggregating**) utilizează aceste subseturi pentru a obține o idee corectă despre distribuție (set complet). Mărimea subseturilor create pentru poate fi mai mică decât setul original.

1. Mai multe subseturi de date sunt create din setul de original, selectând observațiile.

2. Un model de bază este creat pe fiecare dintre aceste subseturi.

3. Modelele funcționează în paralel și sunt independente unul de celălalt.

4. Predicțiile finale sunt determinate prin combinarea previziunilor din toate modelele.

- **Boosting**

Acesta este un proces secvențial, în care fiecare model curent încearcă să corecteze erorile modelului anterior. Modelele ulterioare depind de modelul anterior. Tehnica urmează pașii următori:

1. Un subset este creat din setul de date original.

2. Inițial, toate punctele de date (*data points* = unități discrete de informație) au ponderi egale.

3. Un model de bază este creat pe acest subset.

4. Acest model este utilizat pentru a face previziuni pe întregul set de date.

5. Erorile sunt calculate folosind valorile reale și valorile prezise.

6. Observațiilor care sunt prezise incorect le sunt acordate ponderi mai mari.

7. Se creează un alt model și se fac previziuni pe setul de date. (Acest model încearcă să corecteze erorile din modelul anterior)

8. În mod similar, sunt create mai multe modele, fiecare corectând erorile modelului anterior.

9. Modelul final este media ponderată a tuturor modelelor.

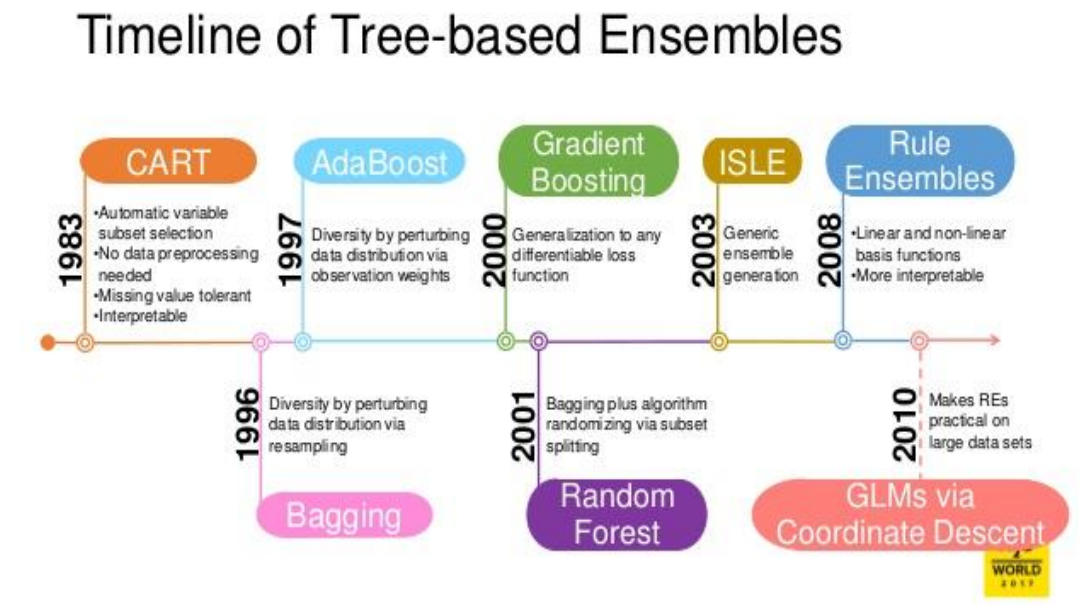


**Figura 6 - Evoluția tehnicilor de *ensemble learning***

## 2.4 Random Forest

### 2.4.1 Generalități

*"**Random forests*** sau ***Random decision forests*** sunt o metodă de **ensemble learning** pentru clasificare, regresie sau alte sarcini, care funcționează prin construirea unei multitudini de arbori de decizie în timpul instruirii scoțând ca *output* clasa ce reprezintă tipul claselor pentru **clasificare**, respectiv predicția pentru **regresie**. ***Random forests*** corectează obiceiul arborilor decizionali de a supra-potrivi (*overfitting*) setul de date de antrenament."

**Random Forest** este un algoritm de **învățare supervizată**, flexibil, care produce, chiar și fără reglarea hiper-parametrilor, un rezultat excelent, de cele mai multe ori. Este, de asemenea, unul dintre algoritmii cei mai folosiți datorită faptului că poate fi folosit atât pentru sarcini de **clasificare**, cât și pentru sarcini de **regresie**.

"Pădurea de arbori" pe care o construiește algoritmul este, de fapt, un ansamblu de arbori de decizie, de cele mai multe ori instruiți prin metoda "***bagging***". ***Bagging***-ul este o abreviere pentru "***bootstrap aggregating***". Acesta este un meta-algoritm, care ia submulțimi M din setul de date inițial și antrenează modelul predictiv pentru acele subsubiecte. Modelul final este obținut prin medierea modelelor "*bootstrapped*" și oferă de obicei rezultate mai bune.

Dezvoltarea timpurie a noțiunii de "**random forest**" a lui Breiman a fost influențată de opera lui Amit și Geman, care au introdus ideea de a căuta un subset aleatoriu al deciziilor disponibile atunci când se divizează un nod, în contextul creșterii unui singur arbore. Ideea selecției aleatoare a subspațiului de la Ho a influențat, de asemenea, designul **random forests**. În această metodă "se cultivă o pădure" de arbori, iar variația între arbori este introdusă prin proiectarea datelor de antrenament într-un subspațiu aleator ales înainte de montarea fiecărui arbore sau a fiecărui nod. În cele din urmă, ideea optimizării nodurilor randomizate, unde decizia la fiecare nod este selectată printr-o procedură randomizată, mai degrabă decât o optimizare deterministă a fost introdusă pentru prima dată de Dietterich.

**Pe scurt**, **Random Forest** construiește mai mulți arbori de decizie și îi îmbină pentru a obține o predicție mai precisă și mai stabilă.

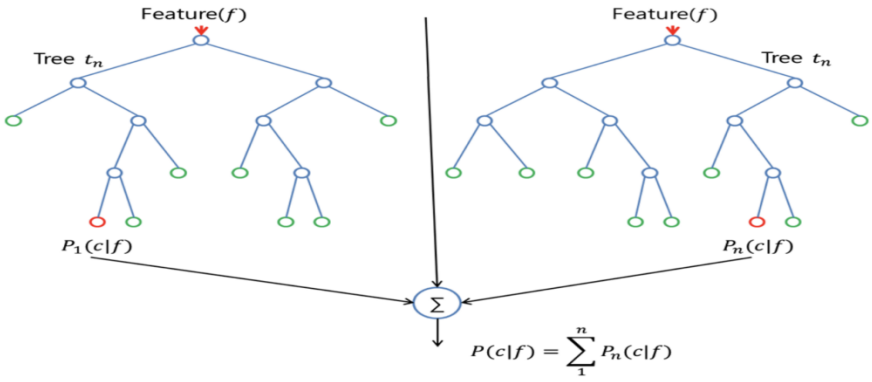


**Figura 7 - Random Forest aplicat pe doi arbori**

Algoritmul de **Random Forest** aduce un plus de "*randomness*" în model, atunci când construiește arborii. În loc să căute cea mai bună caracteristică în timp ce împarte un nod (*splitting*), acesta caută cea mai bună caracteristică într-un subset aleatoriu de caracteristici. Acest proces creează diversitate, care, în general, are ca rezultat un model mai bun.

### 2.4.2 Analogia cu realitatea

Să ne imaginăm o persoană cu nume generic X care vrea să decidă în ce locuri ar trebui să călătorească în timpul unei vacanțe de un an. Aceasta le solicită sugestii cunoscuților. În primul rând, X se duce la un prieten care îl întreabă unde a mai călătorit în trecut și dacă i-a plăcut sau nu. Bazându-se pe răspunsuri (experiența anterioară – *bias*), îi va oferi un sfat lui X.

Aceasta este o abordare tipică a algoritmului **Random Forest**. Prietenul lui X a creat reguli pentru a-și ghida decizia cu privire la ceea ce ar trebui să recomande, folosind răspunsurile lui X.

După aceea, X începe să le ceară din ce în ce mai multor prieteni să-l sfătuiască, aceștia punându-i din nou diferite întrebări, în ideea de a-i oferi unele recomandări. După tot acest proces, X alege destinațiile care i-au fost majoritar recomandate, abordarea tipică a algoritmului **Random Forest**.

### 2.4.3 Diferența între arbori de decizie și *Random Forests*

*Random Forest* este o colecție de arbori de decizie, dar între aceștia există totuși diferențe. Dacă se introduce un set de date de antrenament cu "*labels*"și "*features*" într-un arbore de decizie, acesta va formula un set de reguli care vor fi utilizate pentru a face previziuni.

De exemplu, dacă se dorește anticiparea unui viitor *click* pe o reclamă online, se poate colecta anunțul pe care o persoană a dat *click* în trecut și câteva *features* care descriu decizia sa. Dacă se introduc apoi "*features*" și "*labels*" într-un arbore de decizie, acesta va genera câteva reguli. Apoi se poate prezice dacă pe anunț se va face sau nu *click*. Prin comparație, algoritmul *Random Forest* selectează în mod aleator observații și "*features*" pentru a construi mai mulți arbori de decizie și apoi să medieze rezultatele.

### 2.4.4 Aplicabilitate

*Random Forest* este utilizat într-o mulțime de domenii diferite, cum ar fi sectorul bancar, piața bursieră, medicina și comerțul electronic. În sectorul bancar se utilizează, de exemplu, pentru a detecta clienții care vor utiliza serviciile băncii mai frecvent decât alții, rabursându-și astfel datoria în timp. Mai este utilizat, de asemenea și pentru detectarea clienților cu tentă de fraudă care doresc să înșele banca. În domeniul sănătății se utilizează spre exemplu pentru a analiza istoricul medical al pacientului în ideea de a identifica bolile la care acesta are o predispoziție. În *E-commerce*, algoritmul este folosit pentru a determina dacă un produs este pe placul clientului sau nu.

### 2.4.5 Algoritm

Algoritmul funcționează după cum urmează: pentru fiecare arbore selectăm o mostră din S, unde S (i) reprezintă *bootstrap*-ul cu indicele i. După această etapă antrenăm arborele de decizie folosind un algoritm de învățare de tip arbore de decizie modificat. Acesta se modifică astfel: la fiecare nod al arborelui, în loc să examinăm toate posibilele împărțiri (*split*) de caracteristici (*features*), vom selecta în mod aleator un anumit subset al caracteristicilor $f \subseteq F$, unde F reprezintă setul de caracteristici. Nodul se împarte apoi pe cea mai bună caracteristică în f și nu în F. În practică f este mult, mult mai mic decât F. Decizia privind ce caracteristică să se împartă (*split*) este de cele mai multe ori cel mai scump aspect computațional al arborilor de decizie pentru învățare. Prin îngustarea/reducerea setului de caracteristici (*features*), se accelerează drastic procesul de învățare al arborelui.

**Precondition:** A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features $F$, and number of trees in forest $B$.

```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
function RandomizedTreeLearn(S, F)
    At each node:
        f ← very small subset of F
        Split on best feature in f
    return The learned tree
end function
```

**Figura 8 - Pseudocod Random Forest**

### 2.4.6 "Tuning parameters" pentru modelul Random Forest

Parametrii din *Random Forest* sunt utili fie pentru a mări puterea predictivă a modelului, fie pentru a facilita instruirea acestuia. În funcție de limbajul de programare utilizat, parametrii au diverse titulaturi, aceștia referindu-se însă la aceleași "obiecte"-parte din algoritm. Pentru exemplificare, vor fi analizați în continuare parametrii din limbajul *Python*, cel mai popular pentru aplicațiile de *Machine Learning* datorită librăriilor sale.

- Caracteristici (*features*) ce realizează previziuni mai eficiente

În principiu, există 3 caracteristici ce pot fi "*tuned*" pentru a îmbunătăți puterea predictivă a modelului:

1. **max_features -** nr. maxim de caracteristici pe care le poate încerca *Random Forest* în arborele individual. Există mai multe opțiuni disponibile în Python pentru a seta nr. max de caracteristici. Următoarele exemple sunt câteva dintre ele:
    a. **Auto/None**: Aceasta va lua pur și simplu toate caracteristicile care au sens în fiecare arbore. Nu există nicio restricție asupra arborelui individual.
    b. **Sqrt**: Această opțiune va extrage rădăcina pătrată a numărului total de caracteristici pentru fiecare execuție.

c. **0.2**: Această opțiune permite *Random Forest* să ia 20% din variabilele în execuție individuală. Se poate atribui și valorifica într-un format "0.x", adică x% din caracteristici să fie luate în considerare.

**Ce efect are "ajustarea" max_features asupra performanței și a vitezei ?**

Creșterea parametrilor **max_features** îmbunătățește în general performanța modelului, la fiecare nod având acum un număr mai mare de opțiuni care trebuie luate în considerare. Această creștere scade însă viteza algoritmului. De aceea trebuie păstrat un echilibru odată cu setarea parametrilor.

2. **n_estimators -** Acesta reprezintă numărul de arbori care se dorește a fi construit înainte de a lua votul maxim sau mediile predicțiilor. Un număr mai mare de arbori oferă o performanță mai bună, dar scade viteza de execuție.
3. **min_sample_leaf -** Dacă a fost construit înainte un arbore de decizie, se poate aprecia importanța dimensiunii minime a eșantionului. Frunza (*leaf*) este nodul final al unui arbore de decizie. O frunză mai mică face modelul mai predispus la captarea zgomotului (*noise*) în datele de antrenament.

- Caracteristici (*features*) care influențează viteza de antrenare a modelului:
    1. **n_jobs -** Acest parametru indică nr. de procesoare ce pot fi utilizate. O valoare de "-1" înseamnă că nu există nicio restricție, în timp ce o valoare de "1" înseamnă că se poate folosi doar un singur procesor.
    2. **random_state -** Acest parametru face ca o soluție să fie ușor de replicat. O valoare definită a *random_state* va produce întotdeauna aceleași rezultate dacă îi sunt asignați aceiași parametri și aceleași date de antrenament.
    3. **oob_score -** Aceasta este o metodă aleatorie de validare (*cross validation*) ceva mai rapidă decât altele. Metoda etichetează (*tag*) fiecare observație folosită în diferite momente. Apoi se constată un scor maxim de vot pentru fiecare observație bazată doar pe arbori care nu au folosit această observație specială pentru a se antrena.

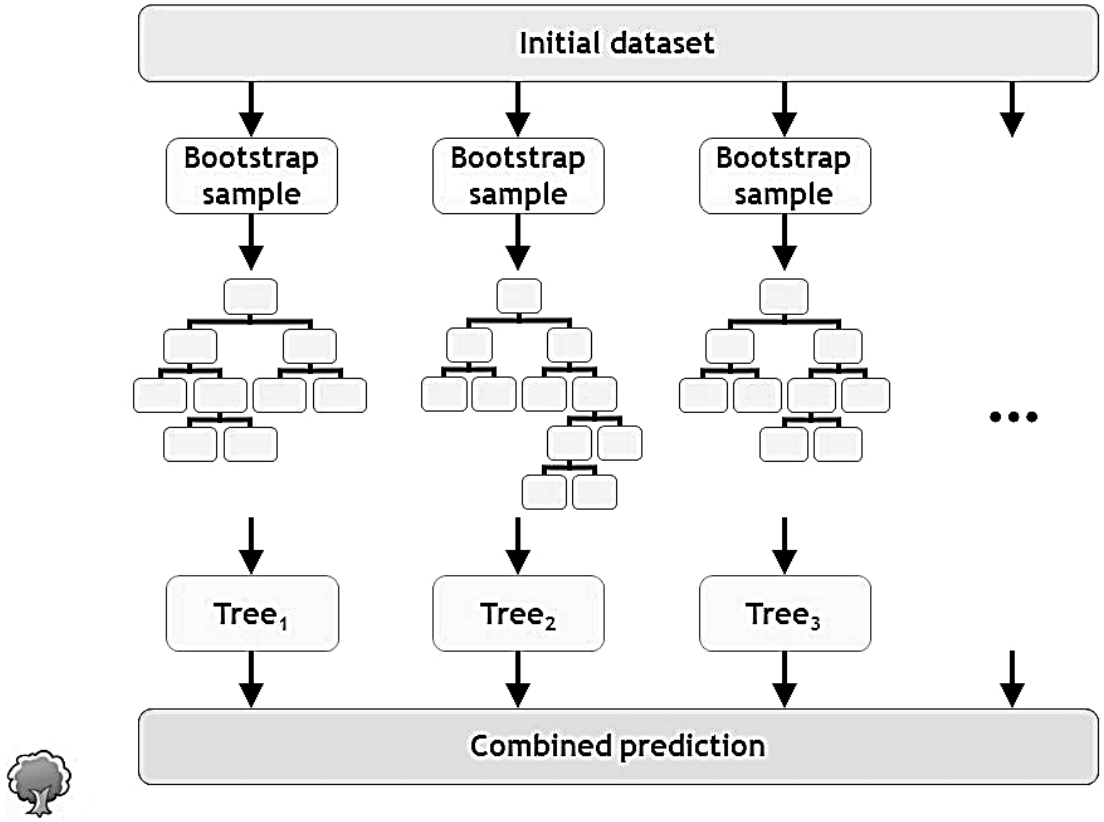


**Figura 9 - Reprezentare grafică simplificată algoritm**

## 2.5 Limbajul de programare JAVA

### 2.6.1 Generalități

**"Java** este un limbaj de programare orientat-obiect, puternic tipizat, conceput de către James Gosling la Sun Microsystems (acum filială Oracle) la începutul anilor '90, fiind lansat în 1995. Cele mai multe aplicații distribuite sunt scrise în Java, iar noile evoluții tehnologice permit utilizarea sa și pe dispozitive mobile gen telefon, agenda electronică, palmtop etc. În felul acesta se creează o platformă unică, la nivelul programatorului, deasupra unui mediu eterogen extrem de diversificat.

Limbajul împrumută o mare parte din sintaxă de la C și C++, dar are un model al obiectelor mai simplu și prezintă mai puține facilități de nivel jos.

Un program Java compilat, corect scris, poate fi rulat fără modificări pe orice platformă pe care este instalată o mașină virtuală Java ( *Java Virtual Machine*, prescurtat JVM). Acest nivel de portabilitate (inexistent pentru limbaje mai vechi cum ar fi C) este posibil deoarece sursele Java sunt compilate într-un format *standard* numit cod de octeți (*byte-code*) care este intermediar între codul mașină (dependent de tipul calculatorului) și codul sursă.

Mașina virtuală Java este mediul în care se execută programele Java. În prezent, există mai mulți furnizori de JVM, printre care Oracle, IBM, Bea, FSF. În 2006, Sun a anunțat că face disponibilă varianta sa de JVM ca open-source."

Un IDE (*integrated development environment*) este un mediu de lucru care permite dezvoltarea de aplicații folosind anumite limbaje de programare (cele suportate de IDE, adică cele pentru care a fost creat acel IDE). Pentru Java sunt folosite următoarele:

- JCreator - gratuit JCreator LE
- Eclipse - gratuit
- NetBeans - gratuit
- BEA Workshop
- BlueJ - gratuit
- CodeGuide - comercial
- DrJava - gratuit
- IntelliJ IDEA - gratuit Idea Community Edition
- JBuilder - comercial
- JDeveloper - comercial, platformă multiplă
- KDevelop - gratuit (platformă GNU/Linux, Cygwin)

### 2.6.2 Caracteristici

Java reprezintă o opțiune populară în cazul alegerii unui limbaj de programare convenabil pentru implementarea diverselor produse software datorită câtorva caracteristici importante pe care le posedă:

- **Simplitate -** Java este uşor de învăţat, caracteristicile complicate (supraîncărcarea operatorilor, moştenirea multiplă, şabloane) întâlnite în alte limbaje de programare sunt eliminate.
- **Robusteţe -** elimină sursele frecvente de erori ce apar în programare prin eliminarea pointerilor, administrarea automată a memoriei şi eliminarea fisurilor de memorie printr-o procedură de colectare a 'surplusului' care rulează în fundal.
- **Complet orientat pe obiecte -** elimină complet stilul de programare procedural, bazându-se pe încapsulare, moştenire, polimorfism.

- **Uşurinţă -** în ceea ce priveşte programarea în reţea
- **Securitate -** este cel mai sigur limbaj de programare disponibil în acest moment, asigurând mecanisme stricte de securitate a programelor concretizate prin: verificarea dinamică a codului pentru detectarea secvenţelor periculoase, impunerea unor reguli stricte pentru rularea programelor lansate pe calculatoare aflate la distanță, etc.
- **Dinamicitate**
- Este **case-sensitive**.
- Permite programarea **multithread (cu fire de execuție)**.
- Este **modelat după C şi C++**, trecerea de la C/C++ la Java făcându-se foarte uşor.
- Permite **dezvoltarea aplicaţiilor pentru Internet -** crearea unor documente Web îmbunătăţite cu animaţie şi multimedia.
- Este **neutru din punct de vedere architectural**.
- **Portabilitate -** Java este un limbaj independent de platforma de lucru, aceeaşi aplicaţie rulând, fără nici o modificare, pe sisteme diferite cum ar fi Windows/UNIX, ceea ce aduce economii substanţiale firmelor care dezvoltă aplicaţii pentru Internet. Sloganul de bază este: „**Write once, run anywhere**".
- Conţine o **librărie de clase şi interfeţe** pentru domenii specifice cum ar fi programarea interfeţelor utilizator (JFC, AWT, Swing), programare distribuită (comunicare TCP/IP, etc.)
- Java Development Kit (JDK) este **disponibil gratis.**

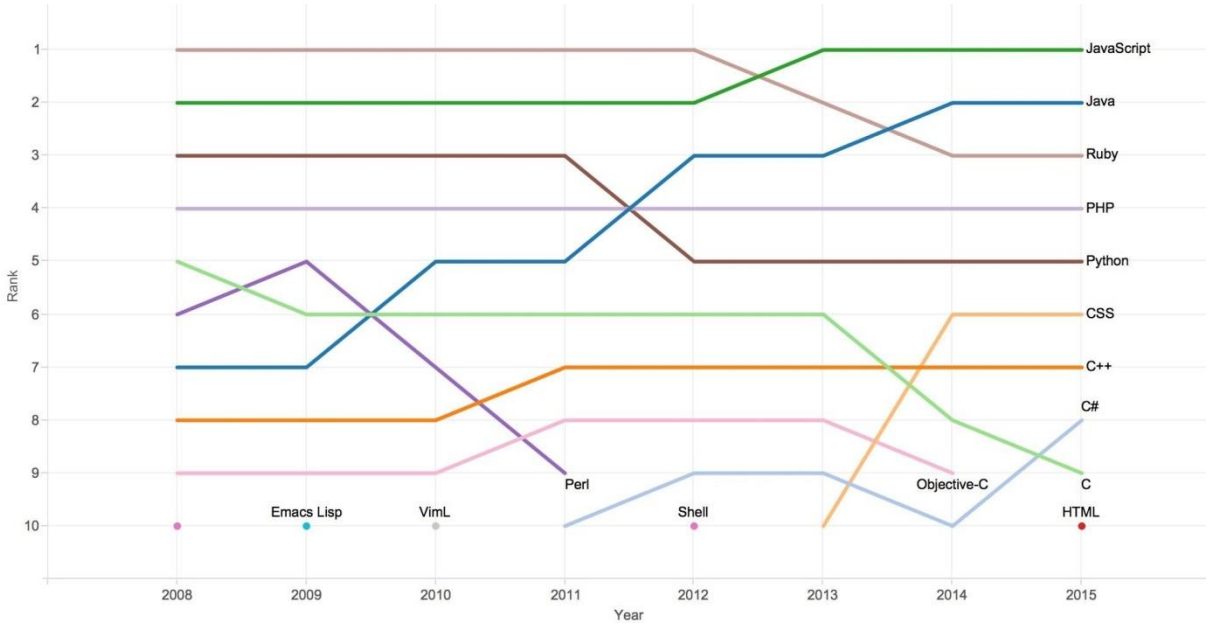


**Figura 10 - Clasament limbaje de programare conform Github.com**

## 2.6.3 Tipuri de limbaje de programare

După modul de execuție al programelor se disting două categorii de limbaje de programare:

**Limbaje interpretate** în care instrucţiunile sunt citite linie cu linie de un program numit interpretor şi traduse în instrucţiuni maşină. Avantajul pe care îl prezintă această categorie este simplitatea, iar ca dezavantaj se remarcă viteza de execuţie redusă.

**Limbaje compilate** în care codul sursă al programelor este transformat de compilator într-un cod numit cod-mașină ce poate fi executat direct de procesor. Avantajul pe care îl prezintă acestă categorie este execuţia rapidă, iar ca dezavantaj se remarcă lipsa portabilităţii, codul compilat într-un format de nivel scăzut neputând fi rulat decât pe platforma pe care a fost compilat.

Programele scrise în limbajul Java sunt atât interpretate cât și compilate, ceea ce reprezintă un atu în implementarea diverselor aplicații/soluții software. În cazul Java, există deci atât un **compilator** care traduce textul unui program scris într-un limbaj de programare „sursă” într-un alt limbaj, numit limbaj

„țintă” /cod de octeți (cod sursă ~ cod obiect), cât și un **interpretor** care citește, analizează și execută codul de octeți.

Codurile de octeţi sunt seturi de instrucţiuni similare codului scris în limbaj de asamblare, fiind generate de compilator. Diferența principală constă în modul de execuție, și anume, **codul mașină** (reprezentat de succesiuni de 0 și 1) este executat direct de procesor, putând fi folosit doar pe platforma pe care a fost creat, în timp ce **codul de octeți** este interpretat de Java, putând fi rulat pe orice platformă ce posedă ca mediu de execuție Java.

Implementarea practică a conceptului independenţei de platformă s-a realizat prin folosirea unui calculator virtual pe care rulează de fapt aplicaţiile compilate java. În urma compilării fişierelor sursă java nu se va genera cod executabil, pentru o platformă anume, ci se va genera un cod intermediar numit *cod de octeţi* (**byte code**). Acest cod de octeţi este asemănător cu limbajul de asamblare dar nu va putea fi rulat direct pe nici un sistem de operare.

### 2.6.4 Mașina virtuală (JVM) și platforme JAVA

Pentru **a rula** un *cod executabil Java* (**cod de octeţi)** este nevoie de un program special care va interpreta acest cod de octeţi şi va executa instrucţiuni specifice sistemului de operare pe care se află. Acest program care interpretează codul de octeţi se numeşte Maşina Virtuală Java.

Maşina Virtuală Java reprezintă un calculator abstract. Ca şi calculatoarele reale, aceasta dispune de un set de instrucţiuni, un set de regiştri şi utilizează diferite zone de memorie.

Platforma Java reprezintă o suită de programe care facilitează implementarea și rularea programelor scrise în Java. O platformă Java include o mașină virtuală, un compilator și un set de librării. DE asemenea, mai poate conține servere și librării adiționale, în funcție de cerințele aplicației software. Java nu este specifică unui anumit procesor sau sistem de operare, platformele de acest tip fiind implementate pentru o mare varietate de hardware și sisteme de operare.

Pentru diverse domenii de aplicabilitate și clase de dispozitive există platforme diferite:

- **Java Card -** O tehnologie care permite aplicațiilor Java de dimensiuni reduse (*applets*) să fie rulate în siguranță pe *smart-cards* sau alte dispositive de memorie similare.
- **Java ME** (Micro Edition) - Specifică diverse colecții de librării (*profiles*) pentru dispozitive având capacități de memorie limitate. Deseori este folosită pentru dispositive mobile.

- **Java SE** (Standard Edition) - Pentru aplicații de uz general pe PC-uri, servere si alte dispositive similare.
- **Java EE** (Enterprise Edition) - Java SE plus alte API-uri utile pentru aplicații de tip client-server.(eventual distribuite)

## 2.6 Apache Tomcat

### 2.6.1 Generalități

*"Apache Tomcat*, cunoscut drept *Tomcat Server* este un container web, *open-source* de Java (*Java Servlet Container*) dezvoltat de Apache Software Foundation (ASF). Un **container web** (*servlet container*) este o componentă a unui web-server care interacționează cu **Java Servlets.** Acesta gestionează ciclul de viață al *servlets,* mai precis, mapează un URL la un anumit *servlet,* asigurându-se că URL-ul are drepturile de acces corespunzătoare.

Un **servlet** este o clasă scrisă în limbajul Java al cărei scop este generarea dinamică de date într-un server HTTP. O astfel de clasă poate crea atât conținut HTML, cât și documente XML, imagini, alte fișiere etc. În cea mai mare parte servleții sunt folosiți împreună cu protocolul HTTP, deși există posibilitatea de a utiliza servleți generici care nu sunt legați de un anumit protocol. Așadar prin „servlet" se înțelege de obicei „servlet HTTP"."

Un servlet nu poate fi executat direct de către server, ci de către un container web. Pentru a răspunde unei cereri de la un client (de obicei un browser) către un anumit servlet, serverul trimite mai departe cererea către container, care se ocupă de instanțierea obiectului respectiv și de apelarea metodelor necesare.

### 2.6.2 Componente

1. **Catalina** – Este *servlet*-ul *container* al serverului Tomcat. Acesta implementează specificațiile *Sun Microsystems* pentru servlet și JSP. În Tomcat, un element *Realm* reprezintă o "bază de date" ce conține nume de utilizatori, parole, roluri asignate acelor *useri*. Diverse implementări ale *Realm* permit **Catalinei** să fie integrată în medii în care informația este deja creată și menținută, ca apoi să fie utilizată la implementarea *Container Managed Security.*

2. **Coyote –** Este componenta de conectare pentru Tomcat (*Connector*) care suportă protocolul HTTP 1.1 ca server web. Acesta îi permite **Catalinei** să se comporte ca un simplu server local, furnizor de documente HTTP.

3. **Jasper** – Este motorul JSP al Tomcat (*JSP Engine*). Jasper parsează fișierele JSP pentru a le putea compila în cod Java sub formă de *servlets* (pentru a fi preluate de **Catalina**). La *runtime*, Jasper detectează modificările apărute în fișierele JSP și le recompilează.

## 2.7 HTML

"HTML (**HyperText Markup Language**) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser.

HTML este o formă de marcare orientată către prezentarea documentelor text pe o singură pagină, utilizând un software de redare specializat, numit *agent utilizator HTML*, cel mai bun exemplu de astfel de software fiind *browserul web*. HTML furnizează mijloacele prin care conținutul unui document poate fi adnotat cu diverse tipuri de metadate și indicații de redare. Indicațiile de redare pot varia de la decorațiuni minore ale textului, cum ar fi specificarea faptului că un anumit cuvânt trebuie subliniat sau că o imagine trebuie introdusă, până la scripturi sofisticate, hărți de imagini și formulare. Metadatele pot include informații despre titlul și autorul documentului, informații structurale despre cum este împărțit documentul în diferite segmente, paragrafe, liste, titluri etc. și informații cruciale care permit ca documentul să poată fi legat de alte documente pentru a forma astfel hiperlink-uri (sau web-ul)."

Componența unui document HTML este:

1. versiunea HTML a documentului
2. zona *head* cu etichetele <head> </head>
3. zona *body* cu etichetele <body> </body> sau <frameset> </frameset>

## 2.8 CSS

"CSS (**Cascading Style Sheets**) este un standard pentru formatarea elementelor unui document HTML. Stilurile se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului, prin elementul `<style>` și/sau atributul `<style>`.

CSS a fost proiectat să permită separarea prezentării de conținut din punct de vedere al aranjării în pagină, culorilor, fontului, etc. Această separare îmbunătățește accesibilitatea conținutului, furnizând mai mult control și flexibilitate în ceea ce privește specificațiile caracteristicilor din prezentare. De asemenea, permite unor pagini web multiple să împărtășească același format, specificând caracteristicile CSS într-un fișier .css separat, reducând astfel complexitatea și repetarea în conținutul structural."

Beneficiile sintaxei **CSS** sunt:

- formatarea este introdusă într-un singur fișier pentru tot documentul
- editarea rapidă a etichetelor
- datorită introducerii într-un singur fișier a etichetelor, se obține o eficientizare a codului paginii, implicit încarcarea mai rapidă a acesteia

Sintaxa **CSS** este structurată pe 3 niveluri:

- nivelul 1 - proprietățile etichetelor din documentul HTML, tip **inline**
- nivelul 2 - informația introdusă în blocul HEAD, tip **embedded**
- nivelul 3 - comenzile aflate în pagini separate, tip **externe**

Importanța sintaxei pornește descrescător de la nivelul 1, acesta fiind prioritar (suprascrie orice alt nivel).

## 2.9 BOOTSTRAP

**Bootstrap** este o librărie *open-source* de *front-end* utilizată pentru designul website-urilor și al aplicațiilor web. Conține diverse template-uri bazate pe tehnologii HTML, CSS, de tipul butoanelor, formularelor și al altor componente caracteristice de interfață. Spre deosebire de alte *framework*-uri, Bootstrap se ocupă exclusiv de partea de *front-end* a aplicațiilor web.

**Bootstrap** este modular și constă într-o serie de *Less stylesheets*, ce implementează diverse componente ale *toolkit*-ului. În general, aceste stylesheets sunt compilate într-un *bundle* (pachet) și incluse în pagini web, alte componente individuale putând fi adăugate, respectiv excluse. În plus față de elementele de tip HMTL, Bootstrap conține diferite elemente frecvent utilizate pentru dezvoltarea *front-end*-ului (*templates*). Componentele sunt implementate ca și clase de tip CSS ce urmează a fi aplicate unor elemente HTML din pagina web. Un avantaj al Bootstrap îl constituie caracteristica de „reusability".

***Bootstrap Grid System*** reprezintă un sistem de *grid*-uri (tabele) *responsive*, fluide, customizabile pentru diverse tipuri de ecrane: telefoane mobile, tablete, etc., cu posibilitatea de expandare pentru laptopuri, PC-uri.

## 2.10 JAVA SERVER PAGES (JSP)

**"JSP** este o tehnologie care permite descrierea unei aplicaţii Web distribuite, specificând într-un singur document atât partea client cât şi partea server. Sintaxa scrierii permite separarea părţii dinamice de partea statică a unei astfel de pagini Web. Partea statică se scrie ca şi un fişier normal HTML sau XML care urmează a fi invocat de browser. Secvenţele dinamice sunt distribuite în acelaşi fişier, intercalate cu elementele din partea statică. Separarea secvenţelor dinamice se face cu nişte tag-uri speciale care conţin construcţii de forma <% ... %> cunoscute şi sub numele de *elemente scripting*. Din punct de vedere al tehnologiilor Java, JSP este o extensie a tehnologiei servlet prin care se permite elaborarea de aplicaţii în stil HTML sau XML."

Evident, nu este unica tehnologie care permite acest lucru. Specificaţiile HTML permit specificarea în documente HTML, prin SSI (Server Side Included), a unor elemente, relativ simple, de comportare a server-ului. Însă cele mai apropiate "rude" ale JSP sunt tehnologiile ASP (Application Server Pages) şi PHP (Programmable/Professional Hypertext Processor după ce iniţial a fost Personal Home Page). Atât ASP cât şi PHP utilizează specificarea intercalată: dinamic - static prin elemente de scripting.

JSP-urile au două **avantaje**:

**Primul** este acela ca partea dinamică este scrisă în Java și nu în Visual Basic sau alt limbaj Microsoft, prin urmare este mai ușor de utilizat.

**Al doilea avantaj** este acela că este portabil în alte sisteme de operare și alte servere diferite de cele Microsoft.

**Tehnologia JSP** este strâns legată de **tehnologia servlet**. **Containerul de *servlet*** asigură toate funcţiile necesare lucrului cu JSP. O pagină JSP este creată într-un fişier de tip .jsp şi este plasată în zona $WEBAPP_NAME - contextul aplicaţiei în containerul de servlet. Pe post de container poate fi Tomcat sau orice alt container servlet. La prima invocare a unei pagini JSP, componenta JSP-Engine preia fişierul .jsp şi construieşte din el un text sursă Java al unui servlet.

Dacă **servlet**-urile sunt clase care generează documente Web, JSP-urile sunt documente web care conțin secvente de cod Java. Prin urmare, JSP-urile sunt considerate ca o extensie a paginilor web obișnuite.

**Tehnologia JSP** permite definirea de biblioteci de noi, tag-uri de către programator. Aceste biblioteci pot fi utilizate ulterior și în cadrul altor aplicații. Mai mult, separă rolurile programatorului și designerului: **programatorul** va stabili care sunt tag-urile, iar **designerul** le va include in documentatia web și nu va mai lucra cu cod Java. Astfel se izolează partea de funcționalitate (motorul aplicației) de partea de vizualizare (interfața cu utilizatorul).

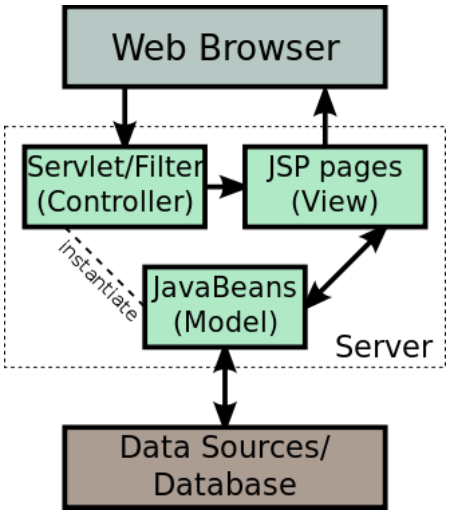


**Figura 11 - Arhitectura MVC (Java)**

## 2.11 Eclipse

**Eclipse** este un mediu de dezvoltare integrat (IDE), utilizat în programare drept cel mai popular IDE de Java. Conține un spațiu de lucru de bază și un sistem extensibil de *plug-in*-uri util pentru customizare. Este scris majoritar în Java și folosit, în principiu, pentru scrierea aplicațiilor în Java, pentru aplicații în alte limbaje de programare precum C, C++, PERL, PHP, RUBY, R, PYTHON, PROLOG, etc., fiind necesare *plug-in*-urile adiționale. Versiunea originală provine de la *IBM VisualAge*.

Kitul de dezvoltare software Eclipse (SDK) care include *tool*-urile de dezvoltare Java este util pentru dezvoltatorii de aplicații Java, dar aceștia își pot extinde abilitățile de programare prin instalarea unor *toolkit*-uri adiționale pentru alte limbaje, dezvoltate pentru această platformă *open-source*.

Diferitelor versiuni de Eclipse li s-au acordat, de-a lungul timpului, diverse nume *science-related*, de la Callisto, Europa și Ganymede, luni ale lui Jupiter, la Helios, Indigo, Juno, Kepler, Luna, Mars, Neon, respective Oxygen.

Platforma Eclipse Web Tools (WTP) este o extensie a platformei Eclipse ce posedă instrumente utile dezvoltării de aplicații web și Java EE. Aceasta include și editoare grafice pentru o varietate de limbi străine, precum și aplicații native (*built-in*) și *API*-uri care susțin dezvoltarea, rularea și testarea programelor implementate.

## 2.12 WEKA

### 2.12.1 Generalități

**"Waikato Environment for Knowledge Analysis** (WEKA) este o colecție de algoritmi de *Machine Learning* scrisă în Java, dezvoltată la Universitatea *Waikato* din Noua Zeelandă în scopuri didactice. Aceasta conține o serie de instrumente de vizualizare și algoritmi utili pentru analiza datelor (*data analysis*) și modelarea predicțiilor (*predictive modeling*), alături de o suită de interfețe grafice care permit utilizatorului sa îi acceseze funcțiile într-un mod facil. Weka este software *open-source*, lansat sub licența publică generală GNU.

WEKA este extensibil și a devenit o colecție de algoritmi pentru învățare în scopul rezolvării problemelor de *data mining* din lumea reală. Fiind implementat în limbajul Java, rulează aproape pe orice platformă. De asemenea, este ușor de folosit din perspectiva utilizatorului (*user-friendly*) și de aplicat pe mai multe niveluri diferite. Librăria WEKA poate fi accesată de propriul program Java, putând implementa și noi algoritmi de învățare.

**Versiunea originală**, non-Java a Weka a fost un *front-end* de tip *Tcl/Tk* (limbaj de programare dinamic, *high-level*, de uz general) ce se ocupa de modelarea algoritmilor implementați în alte limbaje de programare plus alte utilitare pentru preprocesarea datelor scrise în C și un sistem de tip *Makefile-based* pentru a rula diverse experimente de Machine Learning."

**Versiunea actuală**, complet *Java-based*, a cărei dezvoltare a început in 1997, este utilizată în foarte multe domenii, dar, în principiu, în domeniul didactic și cercetare, unde servește ca tool pentru analiza datelor și diverse experimente.

Printre **avantajele** Weka se remarcă:

- Gratuitate sub licența GNU
- Portabilitate, din moment ce este scrisă în Java și rulează pe aproape orice platformă modernă de calcul.
- O colecție vastă și inteligibilă de algoritmi și metode de preprocesare
- Facilitate în executarea sarcinilor datorită interfeței *user-friendly*

### 2.12.2 Interfața cu utilizatorul

Principala interfață Weka este fereastra de *Explorer*, dar, în principiu, aceeași funcționalitate poate fi accesată prin intermediul interfeței *Knowledge Flow* și din linia de comandă. De asemenea, există și fereastra *Experimenter*, ce permite compararea sistematică a performanțelor predictive ale Weka pe co colecție de seturi de date.

Interfața *Explorer* prezintă o serie de *panel*-uri care oferă acces la principalele componente ale *software*-ului:

- Panoul *Preprocess* are diverse facilități pentru importarea datelor dintr-o bază de date, dintr-un fișier .csv, etc. și o preprocesare a datelor denumită "*filtering algorithm*", utilizată pentru selecția atributelor pe diverse criterii.
- Panoul *Classify* care permite aplicarea algoritmilor de clasificare, respectiv regresie pe diferite seturi de date.
- Panoul *Associate* care furnizează o serie de diagrame în ceea ce privește relațiile dintre atribute.
- Panoul *Cluster* care oferă acces la tehnici de *clustering.*
- Panoul *Select attributes* care furnizează algoritmi pentru identificarea celor mai *predictive* atribute din setul de date.
- Panoul *Visualize* care prezintă o matrice de tip "scatter plot"

### 2.12.3 Instrumente native de regresie

Weka posedă un număr relativ mare de instrumente de regresie și clasificare. Pachetele native se regăsesc în executabilul de Weka, iar altele, non-native pot fi descărcate ulterior. Printre instrumentele de regresie se numără:

- M5Rules
- DecisionStump M5P
- RandomForest
- RepTree
- ZeroR
- DecisionRules
- LinearRegression
- SMOreg
- SimpleLinearRegression
- MultiLayerPerceptron
- GaussianProcesses

## 2.13 "KAGGLE: Your home for data science !"

### 2.13.1 Generalități

**Kaggle** este o platformă pentru competiții din domeniul "*predictive modelling & analytics*" (analiză și predicție) unde statisticieni și "*data miners*" concurează pentru a produce cele mai bune modele pentru predicția și descrierea seturilor de date furnizate de diverse companii. Această abordare de tip *crowdsourcing* se bazează pe faptul că există numeroase strategii care pot fi aplicate oricărui *task* de modelare predictivă, fiind imposibil să știi dinainte ce tehnică sau ce analist va fi mai eficient.

Pe data de 8 martie 2017, Google a anunțat că va achiziționa **Kaggle**. Acesta se va alătura echipei *Google Cloud*, urmând să își continue însă activitatea ca brand independent. În luna ianuarie 2018, Booz Allen și **Kaggle** au anunțat lansarea *Data Science Bowl*, o competiție de *Machine Learning* menită să analizeze componența celulelor și să identifice nucleii, o abordare de-a dreptul ambițioasă pentru un domeniu de cercetare.

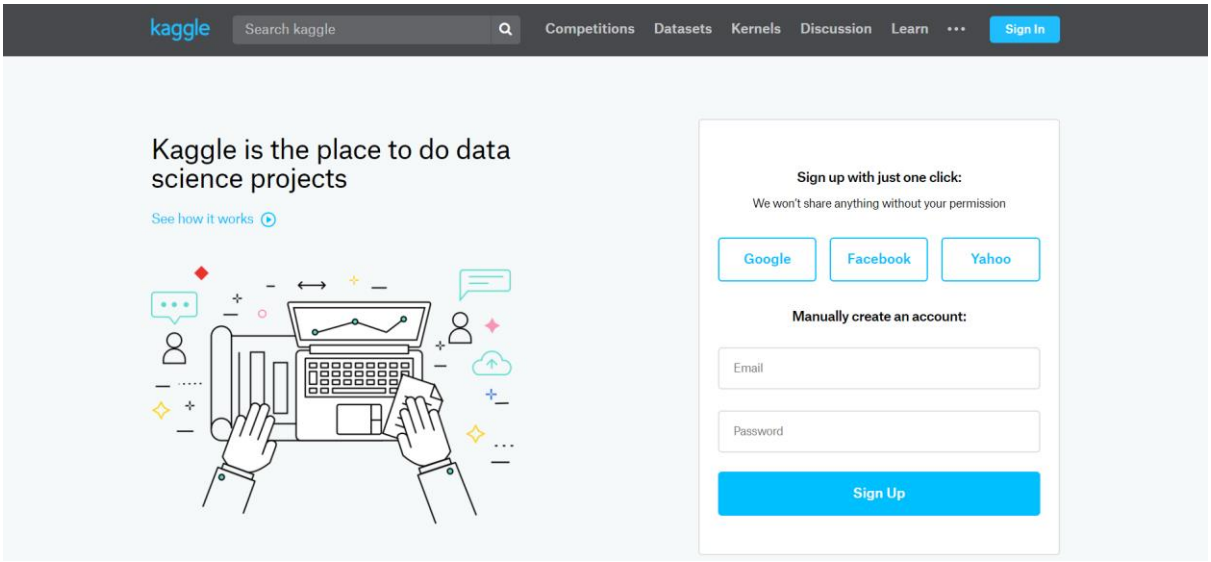


**Figura 12 - Kaggle Homepage**

## 2.13.2 Comunitatea Kaggle

Conform unor statistici efectuate în luna mai 2016, platforma **Kaggle** avea peste 536.000 de utilizatori înregistrați, comunitatea acoperind peste 154 de țări. Comunitatea **Kaggle** este cea mai cunoscută și diversificată din lume în domeniul *data science*, meritându-și pe bună dreptate această titulatură, fapt dovedit de-a lungul timpului. Membrii **Kaggle**, așa numiții *Kagglers*, provin dintr-o varietate de medii profesionale, activând în domenii precum *computer science*, *computer vision* sau chiar medicină, biologie, etc., domenii la care nimeni nu s-ar aștepta, deoarece, aparent, nu au nicio legătură cu zona în care activează **Kaggle**. Fapt cât se poate de neadevărat.

Competițiile **Kaggle** atrag, în mod regulat, peste 1.000 de participanți, individual sau organizați în echipe. Comunitatea este una în permanență activă și devotată muncii și cauzei proprii, dovadă fiind cele peste 4.000 de postări pe lună și peste 3.500 de rezolvări înscrise în competiții pe zi. De asemenea, această "societate" include foarte mulți dintre cei mai cunoscuți și experimentați cercetători în domeniu, chiar membrii ai echipei câștigătoare de la concursul *Jeopardy* - echipa *IBM Watson*, precum și membrii ai echipei de la *Google Deepmind*. Mulți dintre aceștia și-au dovedit performanțele în cadrul competițiilor de pe **Kaggle**.

Ca o paranteză, *proiectul Watson* a fost un adevărat success în domeniul Inteligenței Artificiale și al ML, reușind ca prin mix-ul dintre cele două să aducă în prim plan, impactul fantastic al mașinăriilor inteligente în viața cotidiană, mai precis la o populară emisiune televizată din Statele Unite ale Americii, unde *Watson* a "atras luminile reflectoarelor" asupra sa.

"**Watson** este un sistem computerizat cu inteligență artificială, capabil de a răspunde la întrebări puse în limbaj natural. A fost dezvoltat prin proiectul DeepQA - IBM de către o echipă de cercetare condusă de David Ferrucci. Watson a fost numit după primul președinte IBM, Thomas J. Watson.

În 2011, pentru a se testa abilitățile sale, Watson a concurat în concursul *Jeopardy!*, fiind în acel moment prima înfruntare de acest fel între om și mașină. În trei emisiuni televizate, Watson l-a învins pe Brad Rutter, cel care a câștigat cea mai mare sumă la acest concurs și pe Ken Jennings, cel care a avut cele mai multe emisiuni câștigate (74). Watson a primit marele premiu de 1 milion dolari, în timp ce Ken Jennings și Brad Rutter au primit 300.000 dolari și respectiv 200.000 dolari. Jennings și Rutter s-au angajat să doneze jumătate din câștiguri, în timp ce IBM a împărțit câștigul lui Watson la două organizații de caritate.

Watson în mod constant și-a depășit adversarii umani, dar a avut probleme să răspundă la câteva categorii, în special întrebările care au indicii scurte și care conțin numai câteva cuvinte. Pentru fiecare indiciu, primele trei răspunsuri cele mai probabile ale lui Watson erau afișate pe un ecran de televiziune. Watson a avut acces la 200 de milioane de pagini cu conținut structurat și nestructurat care au însumat 4 Teraocteți de stocare pe disc, inclusiv textul integral al site ului Wikipedia. Watson nu a fost conectat la Internet în timpul jocului." ( conform *Wikipedia*)

### 2.13.3 Cum se desfășoară competițiile Kaggle ?

1. Gazdele competiției pregătesc setul de date și descrierea competiției curente. **Kaggle** oferă un serviciu de consultanță care ajută gazda cu toate aceste pregătiri, cu anonimizarea datelor și integrarea modelului câștigător în operațiunile curente ale companiei.

2. Participanții experimentează cu diverse tehnici și concurează unii împotriva celorlalți cu scopul de a produce cele mai bune modele. Rezultatele muncii lor sunt expuse în mod public prin intermediul secțiunii *Kaggle Scripts*, aceasta devenind astfel o colecție de rezolvări ce poate constitui o bună sursă de inspirație pentru noii competitori. După încărcarea rezultatelor obținute sub un format standard prestabilit, cei de la Kaggle realizează o evaluare (bazată pe acuratețea predictivă obținută de ei pe setul de date soluție/ascuns), efectuând apoi un

clasament *world-wide*, unde fiecare rezolvitor își poate vedea poziționarea soluției proprii, eventual pentru o viitoare îmbunătățire.

3. După încheierea competiției (fiecare competiție are asignat un *deadline*), gazda (compania) plătește un premiu monetar în schimbul soluției câștigătoare. (*"a worldwide, perpetual, irrevocable and royalty free license to use the winning entry."*). De asemenea, după locul ocupat în clasament, rezultatul obținut și descrierea soluției, companiile îi pot contacta pe participanții care le-au atras atenția în mod spcial, spre a-i intervieva în ideea unei posibile angajări.

### 2.13.4 Impactul competițiilor Kaggle

De-a lungul timpului, competițiile au furnizat o mulțime de proiecte de succes, acoperind o arie foarte largă de domenii. De la cercetări în domeniul HIV, predicții în ceea ce privește sportul minții (șah), vreme, trafic, mobilitatea populației, riscuri de accidente, etc. până la diverse lucrări științifice în domeniul academic, **Kaggle** a fost un real suport. Datorită clasamentului public pe care îl actualizează în timp real, Kaggle încurajează competitorii să inoveze când vine vorba de soluții, promovând spiritul academic, dorința de afirmare și pornirile competitive.

**De ce Kaggle și nu altceva ?**

Deși există numeroase modalități de a învăța și experimenta în domeniul *Machine Learning*, Kaggle este una dintre cele mai eficiente platforme suport pentru toate aceste procese, aducând o serie de beneficii deloc de neglijat în fața competitorilor din segmentul în care activează.

- Problemele sunt bine definite, iar seturile de date sunt furnizate direct, fiind colectate și validate în prealabil.

- Clasamentul oferă o viziune clară asupra rezultatelor pe care tu, ca și competitor, le obții, fiind evaluate după un standard (universalitatea evaluării).

- Fiecare competiție oferă drept suport o secțiune de comentarii asupra soluțiilor, dezbateri, schimburi de opinii, din care poți atât învăța, asimila cunoștiințe noi, cât și la care poți contribui prin participare activă.

- Îți poți construi un portofoliu de proiecte cu probleme de actualitate din viața de zi cu zi care vor contribui la dovedirea *skill*-urilor pe care le posezi.

- Este un mediu în care calitatea de competitor/participant aduce în prim-plan conceptul de uniformitate, de pornire de la aceleași premise, cu șanse egale, indiferent de mediul profesional în care activează fiecare.

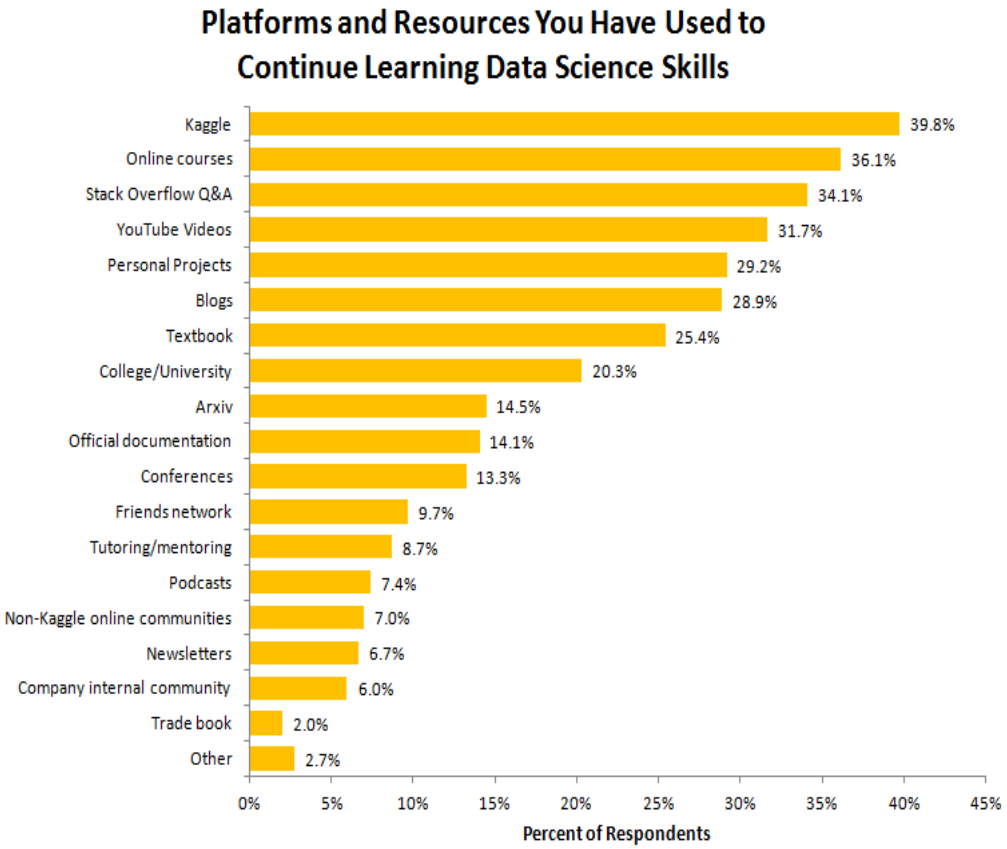


**Figura 13 - Poziția Kaggle în topul platformelor suport pentru învățare în domeniul *data science***

# 3 IMPLEMENTARE APLICAȚIE

## 3.1 Scopul aplicației

Aplicația web reprezintă un instrument util de administrare a unei companii de închirieri biciclete, menită să ofere suport/ghidaj în ceea ce privește luarea deciziilor și alegerea strategiilor de *business* adecvate afacerii. Esența acestei aplicații este constituită de componenta de *Machine Learning*, cu ajutorul căreia se realizează predicții în ceea ce privește numărul de închirieri de biciclete (din viitor) în funcție de "antecedente", mai precis, de un set de date statistice, colectate din trecut care oferă posibilitatea identificării unor diverse corelații, observații, *trend*-uri, etc.

Aplicația este abordată din două perspective (dublu rol):

- *data analistul*, care se va ocupa de preprocesarea datelor, colectarea și validarea seturilor de date, antrenarea modelului, validarea acestuia și obținerea unui rezultat oficial furnizat și confirmat de cei de la Kaggle.
- Proprietarul afacerii, care va avea posibilitatea ca setând numeroși parametrii furnizați de media (exemplu: cantitatea de precipitații, viteza vântului, etc.) să primească o predicție informativă/orientativă pentru o anumită dată selectată, păstrând însă marja de eroare furnizată anterior de Kaggle.

Acest dublu rol se concretizează în două interfețe grafice personalizate, *user-friendly* și adaptate necesităților fiecărui tip de utilizator.

## 3.2 Analiza seturilor de date

Procedura standard de desfășurare a unui task de *Machine Learning* este următoarea:

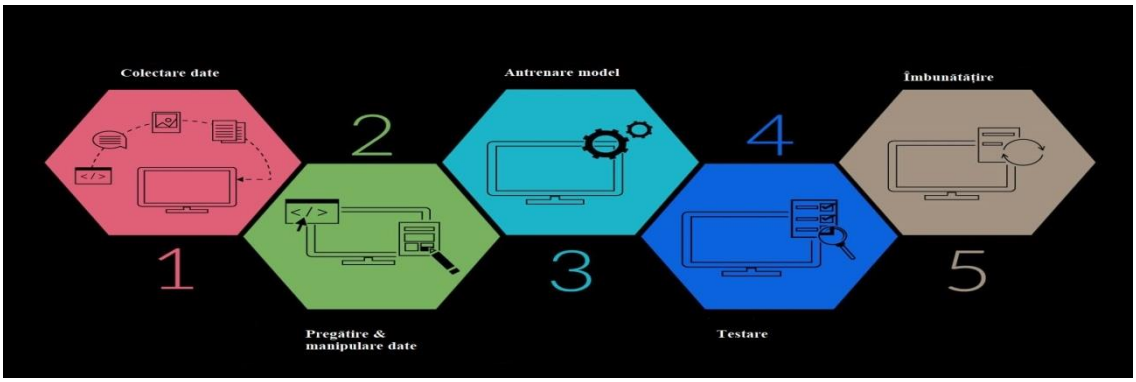


**Figura 14 - Pași în modelarea predicției**

- **Colectare date necesare**

Seturile de date utilizate ca *input* în cadrul aplicației au fost furnizate de platforma Kaggle, date colectate anterior de compania care a propus această competiție. Kaggle s-a ocupat de organizarea seturilor de date în format *.csv*, specific rezolvării acestui tip de *task*-uri (există însă și varianta *.arff* sau *.xls*) și ales ca standard încă de la apariția acestei platforme de *data science*. De asemenea, pentru încărcarea rezultatelor și evaluarea acestora s-a folosit același format *.csv*. (în alt format nu se poate înscrie soluția în competiție și deci, nici nu se poate primi o evaluare/un scor)

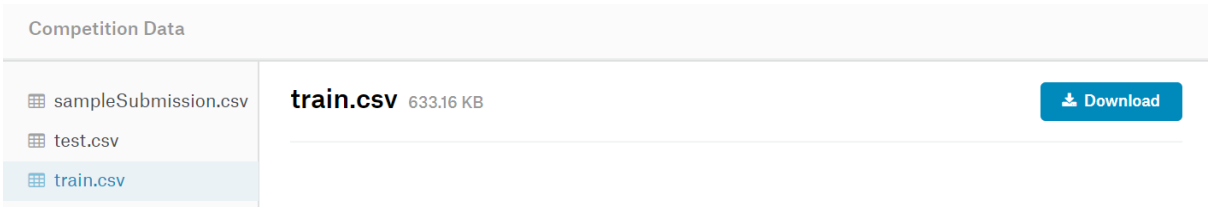


**Figura 15 - Fișier date de antrenament**

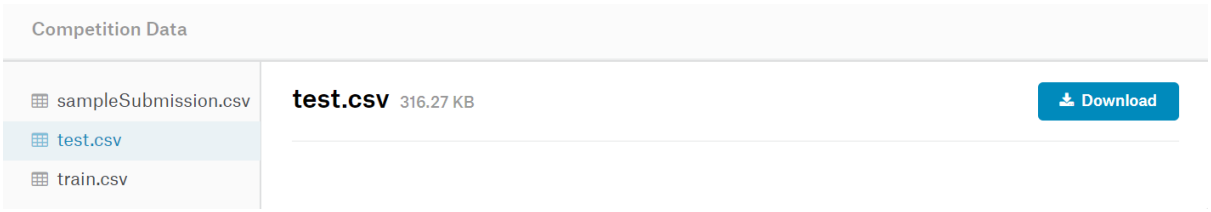


**Figura 16 - Fișier date de test**

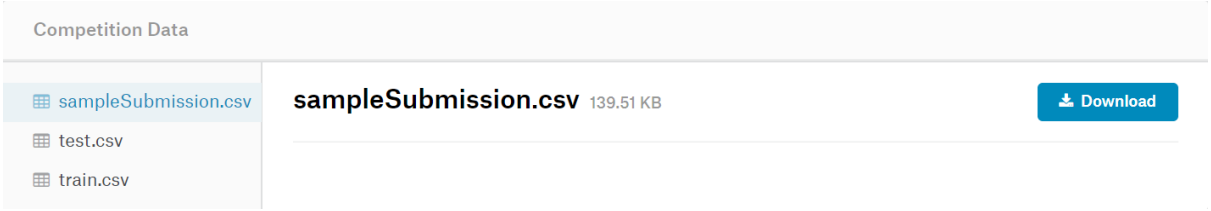


**Figura 17 - Format fișier rezultat**

- **Pregătire & manipulare date**

Este etapa în care se analizează seturile de date, *train.csv* și *test.csv* din punct de vedere al semnificației variabilelor, tipului de date, respectiv al relațiilor dintre acestea. (*trend*-uri, variabile lipsă, etc.)

- *Train.csv*

Este fișierul care conține setul de date de antrenament, utilizat ca input. Datele au fost colectate în urma observațiilor din trecut (1/2 ani în urmă). După antrenament, modelul va fi capabil să realizeze o predicție în ceea ce privește numărul de închirieri de biciclete. Această etapă mai poartă denumirea de **preprocesare**.

Structura *train.csv* este următoarea:

```
datetime,season,holiday,workingday,weather,temp,atemp,humidity,windspeed,casual,registered,count
2011-01-01 00:00:00,1,0,0,1,9.84,14.395,81,0,3,13,16
2011-01-01 01:00:00,1,0,0,1,9.02,13.635,80,0,8,32,40
2011-01-01 02:00:00,1,0,0,1,9.02,13.635,80,0,5,27,32
2011-01-01 03:00:00,1,0,0,1,9.84,14.395,75,0,3,10,13
2011-01-01 04:00:00,1,0,0,1,9.84,14.395,75,0,0,1,1
2011-01-01 05:00:00,1,0,0,2,9.84,12.88,75,6.0032,0,1,1
2011-01-01 06:00:00,1,0,0,1,9.02,13.635,80,0,2,0,2
2011-01-01 07:00:00,1,0,0,1,8.2,12.88,86,0,1,2,3
2011-01-01 08:00:00,1,0,0,1,9.84,14.395,75,0,1,7,8
```

**Figura 18 - Mostră din *train.csv***

În *Machine Learning*, parametrii din setul de date de antrenament se numesc **atribute**.

Semnificația variabilelor din setul de date este următoarea:

| Variabilă | Semnificație |
|---|---|
| **datetime (v.independentă)** | data & ora (*timestamp*) |
| **season (v.independentă)** | 1 = primăvara, 2 = vara, 3 = toamna, 4 = iarna |
| **holiday (v.independentă)** | dacă ziua este considerată ca fiind de vacanță |
| **workingday (v.independentă)** | dacă ziua nu este nici de weekend, nici de vacanță |

| | |
|---|---|
| **temp (v.independentă)** | temperatura în grade Celsius |
| **atemp (v.independentă)** | temperatura “resimțită în grade Celsius |
| **humidity (v.independentă)** | umiditatea relativă |
| **windspeed (v.independentă)** | viteza vântului |
| **casual (v.dependentă)** | nr. de închirieri ocazionale (fără abonament) |
| **registered (v.dependentă)** | nr. de închirieri cu abonament |
| **count (v.dependentă)** | nr. total de închirieri |

**Tabelul 1 - Atributele și semnificația acestora**

Ca și tipuri de variabile, doar *datetime* este o variabilă nominală, restul fiind variabile numerice.

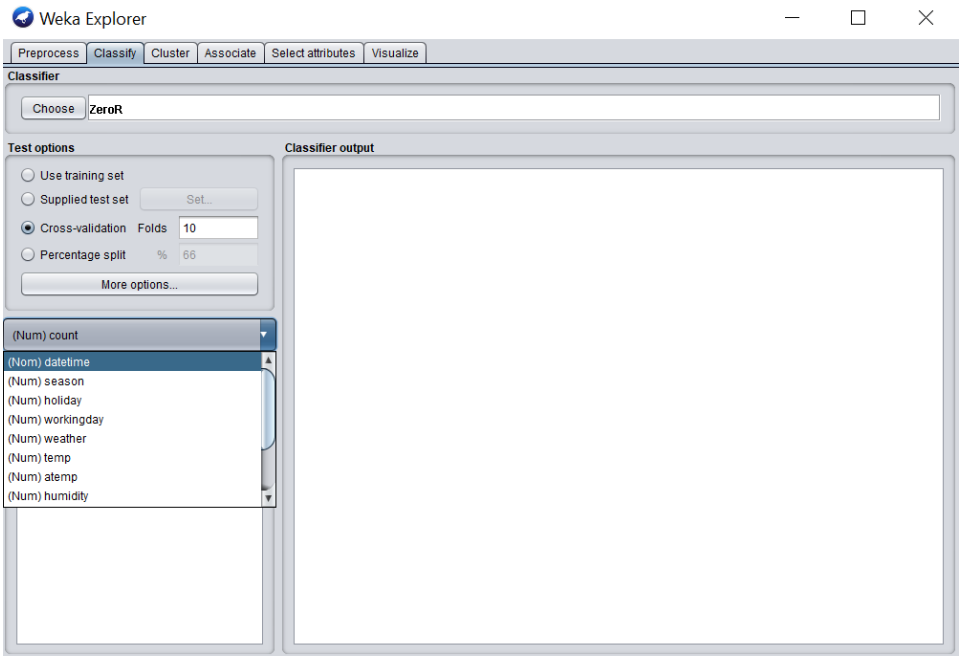


**Figura 19 - Tipul variabilelor (Explorer-ul Weka)**

- Un prim pas după stabilirea semnificației variabilelor/atributelor din setul de date de antrenament este stabilirea valorilor lipsă. (*missing values*) Această verificare este esențială în cazul problemelor de *Machine Learning* pentru a evita viitoare nereguli în ceea ce privește seturile de date.

*Train.csv* a fost încărcat în IDE-ul Weka, unde a fost testat pentru *missing values*. Din fericire, acest set de date este complet.

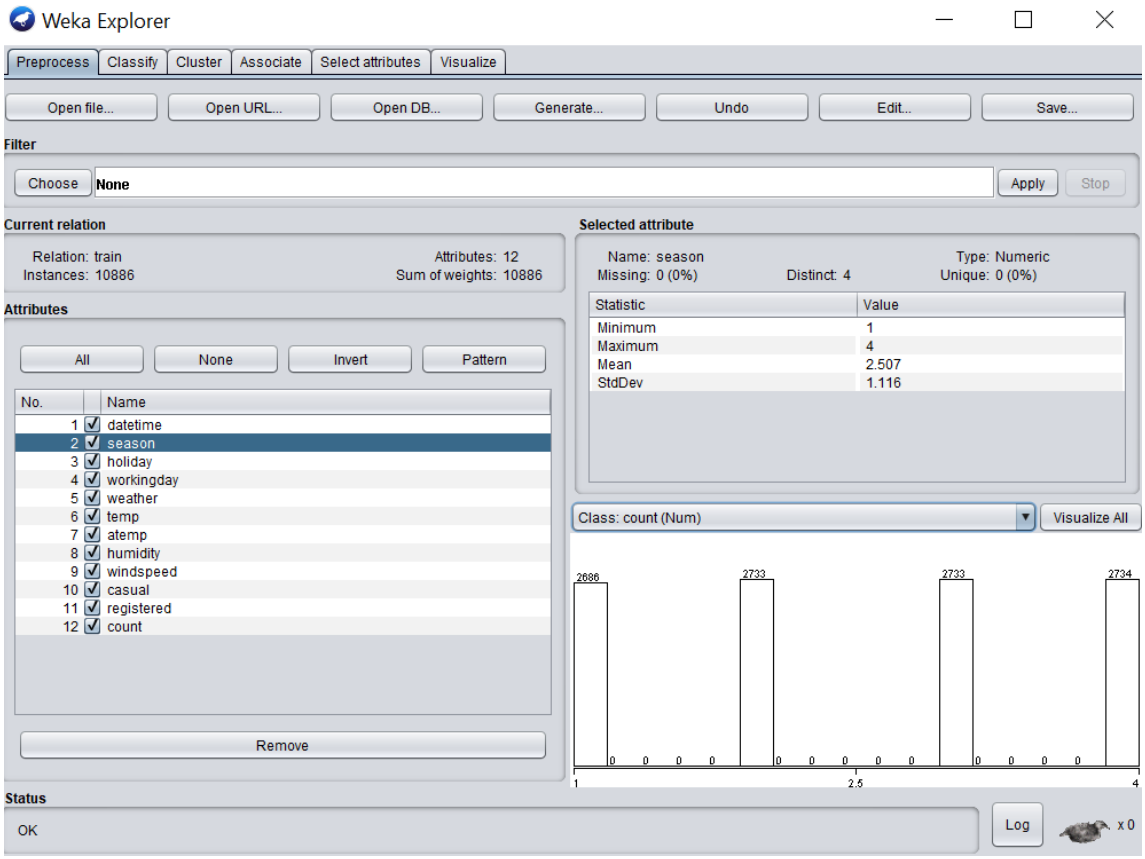


**Figura 20 - Verificare valori lipsă (no missing values)**

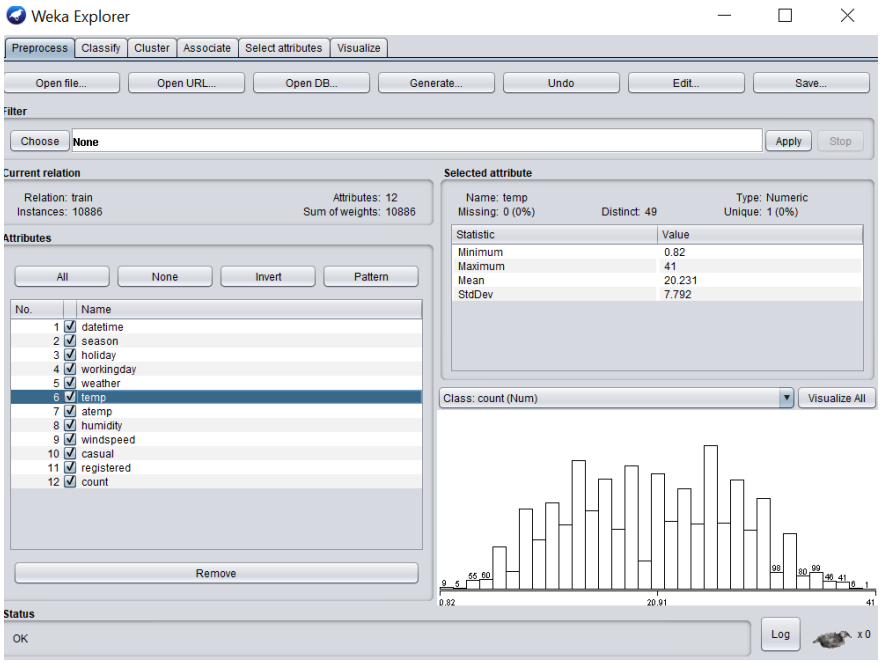


**Figura 21 - Continuare Figura 20**

Pentru întreg setul de date și apoi în particular pentru fiecare atribut, Weka furnizează numărul de atribute, respectiv de instanțe, minimul, maximul valorilor, dar și alte caracteristici utile reprezentării grafice ulterioare a distribuției. (deviația standard, media valorilor, etc.)

- Vizualizarea distribuției atributelor

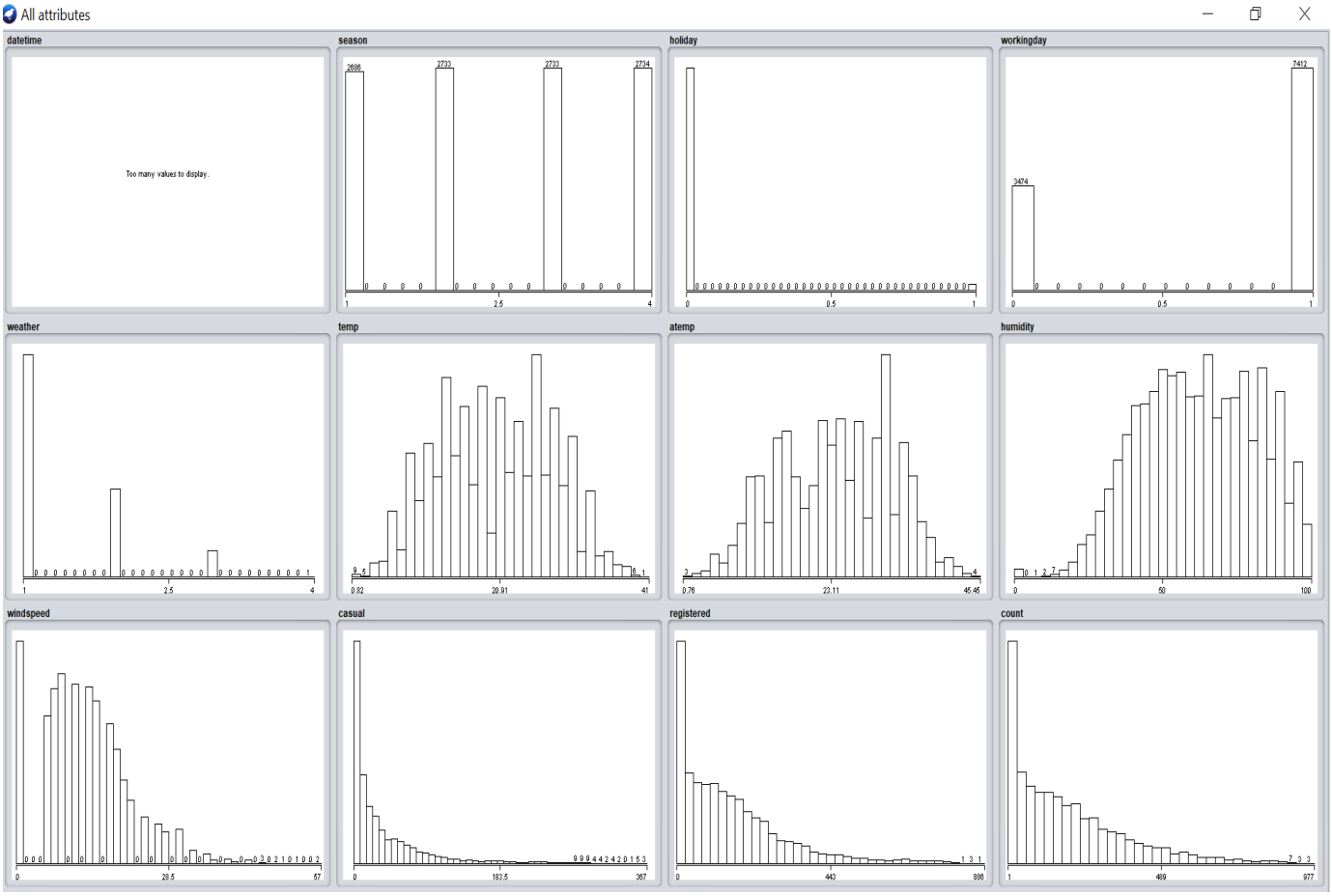


**Figura 22 - Distribuție atribute - instanțe în *train.csv***

- Matricea de "interacțiune" a atributelor (***Scatterplot matrix***)

După evaluarea proprietăților caracteristicilor individuale se pot analiza legăturile formate din diverse combinații de atribute. Weka permite o vizualizare grafică a acestor interdependențe (raporturi între perechi de atribute). Această reprezentare este utilă deoarece evidențiază tiparele/șabloanele în relația dintre atribute.

Pentru un set de variabile de date (dimensiuni) X1, X2, ..., Xk, ***matricea scatterplot*** arată toate *plot*-rile de dispersie pereche a variabilelor dintr-o singură vizualizare cu multiple scatterplot-uri într-un format de matrice. Pentru k variabile, ***matricea scatterplot*** va conține k rânduri și k coloane. Un grafic aflat pe intersecția rândului i și coloanei j este o diagramă a variabilelor Xi versus Xj. Acest lucru înseamnă că fiecare rând și coloană reprezintă câte o dimensiune, iar fiecare celulă va prezenta atunci o scală de două dimensiuni.

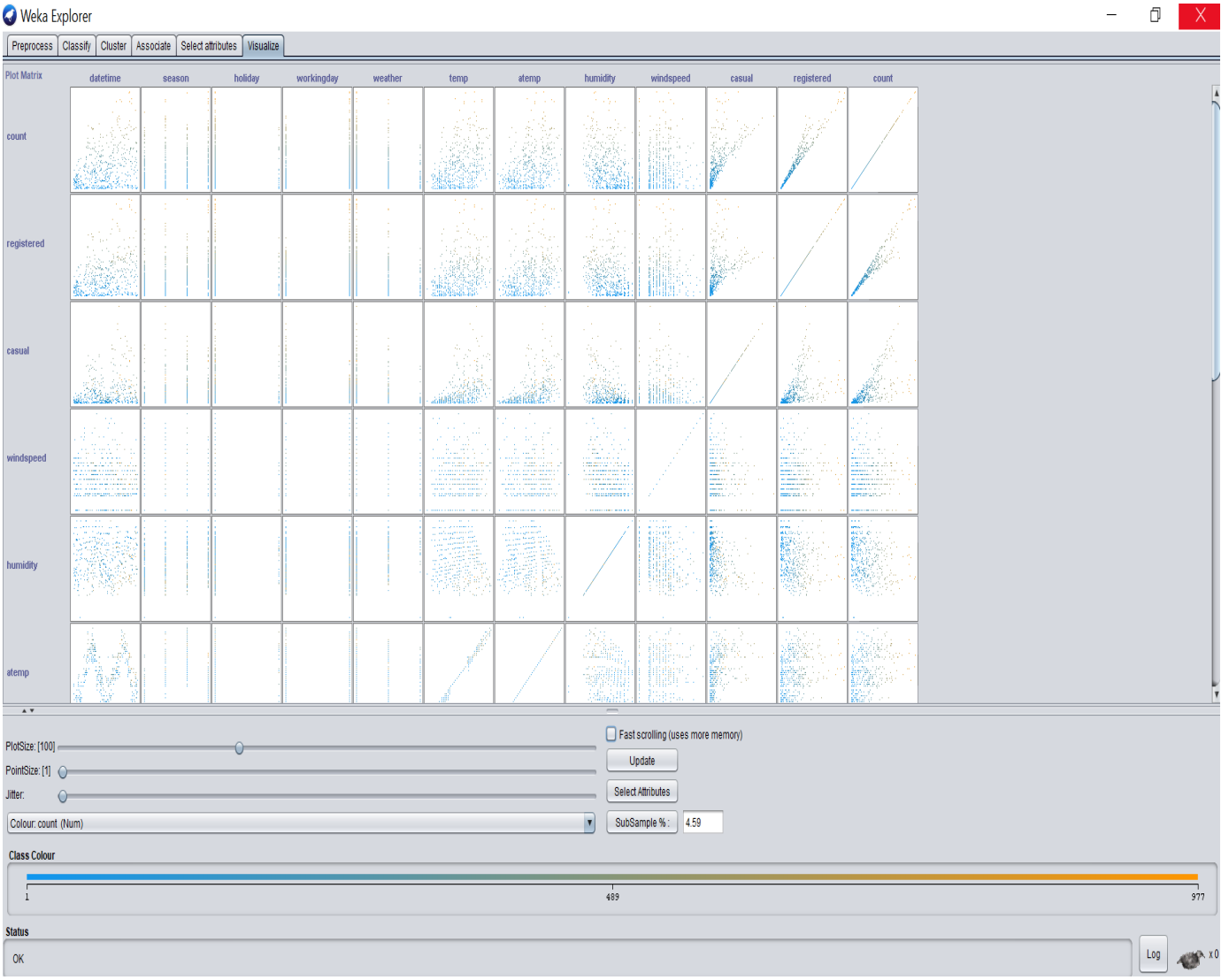


**Figura 23 - *Scatterplot matrix* pentru toate perechile de atribute**

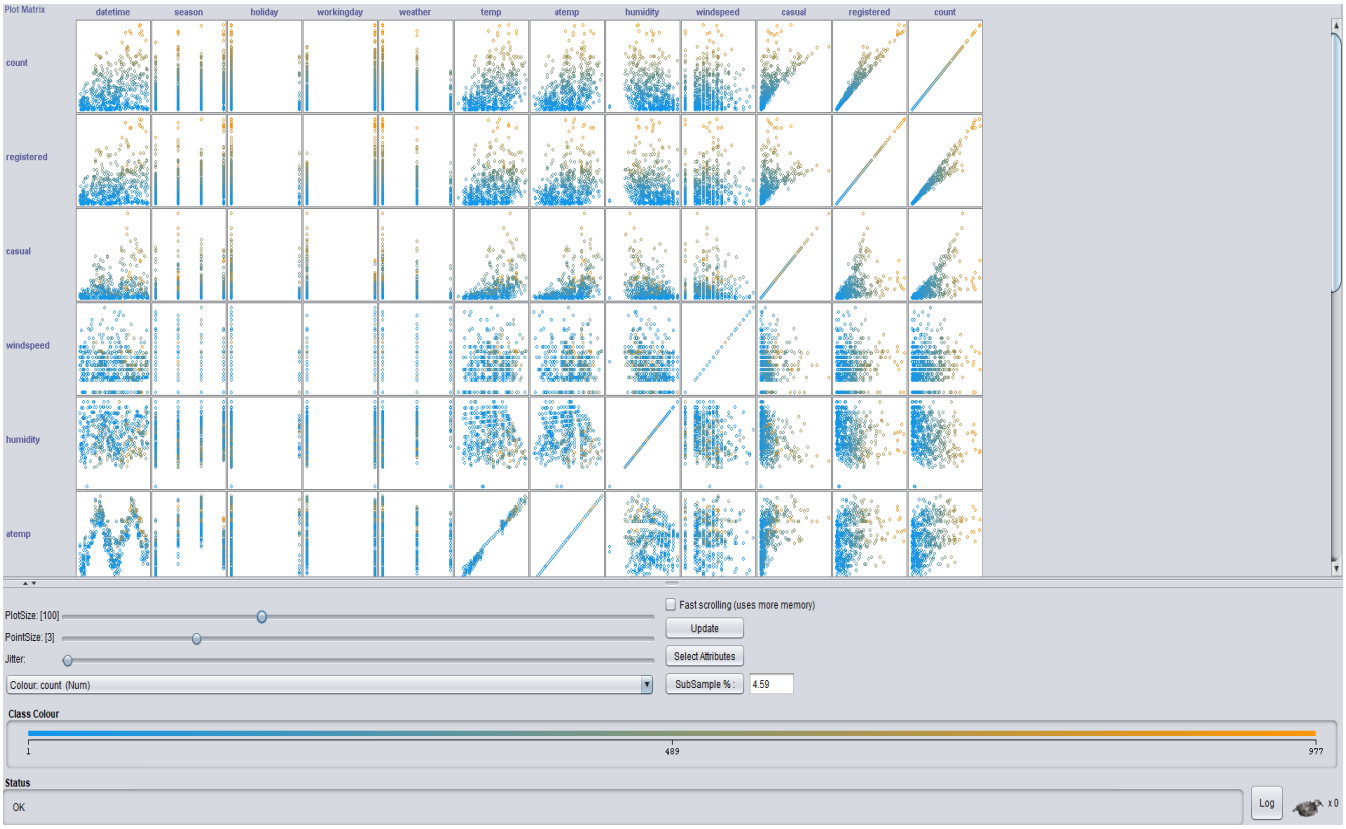


**Figura 24 - *Scatterplot matrix* cu mărirea dimensiunii punctelor (avantaj-vizualizare *trend*-uri)**

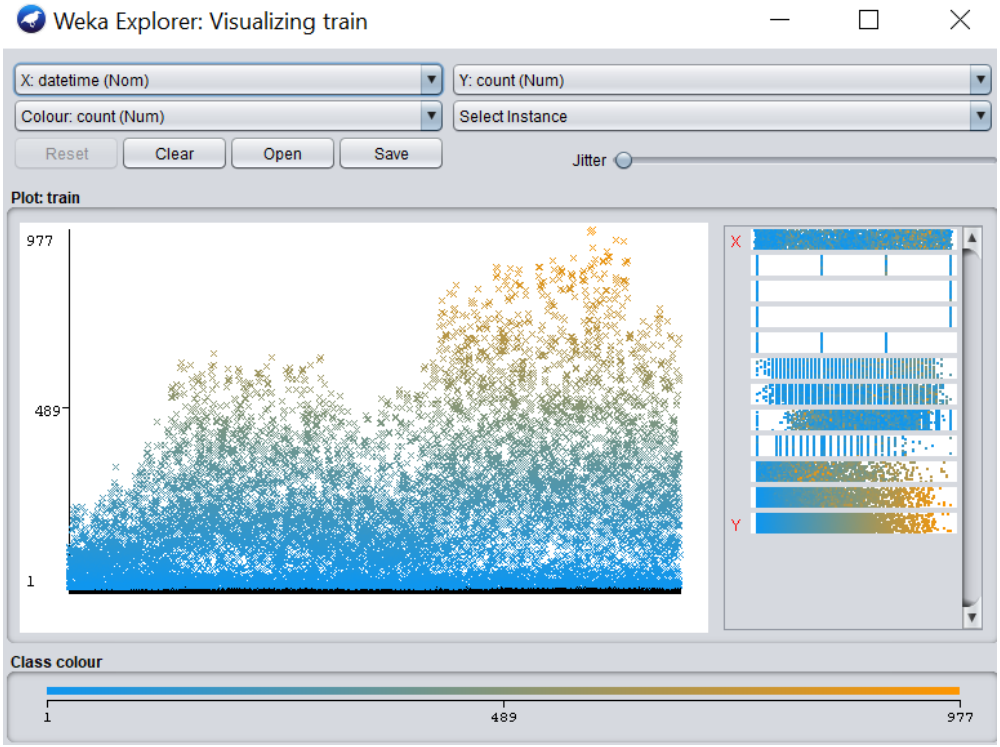


**Figura 25 - *Scatterplot matrix* pentru o pereche de atribute**

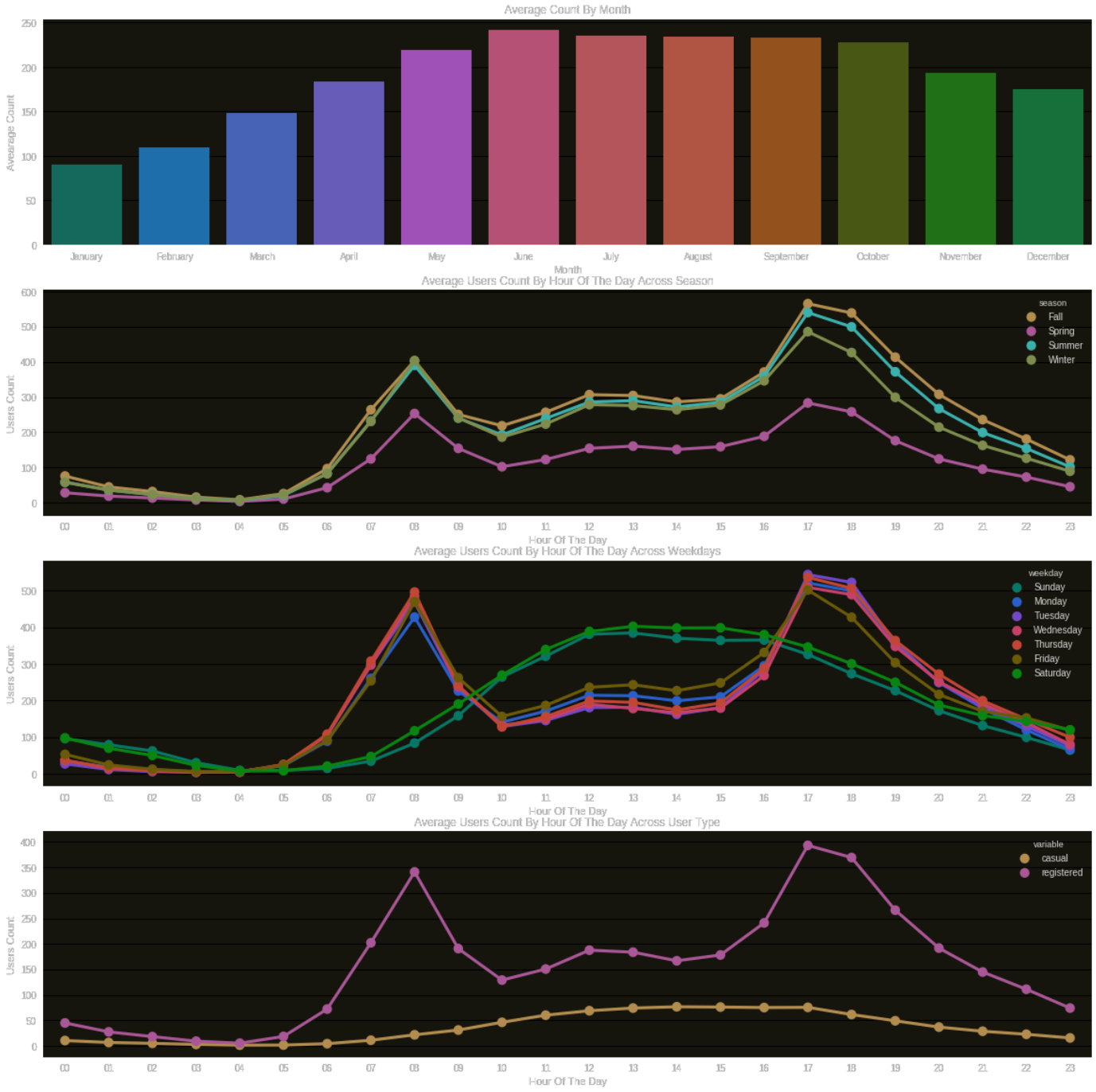


**Figura 26 - Chart-uri distribuție atribute**

Din aceste reprezentări grafice se pot remarca următoarele:

- Oamenii tind să închirieze mai multe biciclete pe timpul verii, ceea ce este absolut normal datorită condițiilor meteo favorabile. Se observă deci că numărul de închirieri este semnificativ mai mare în lunile iunie, iulie, august.

- În zilele lucrătoare, majoritatea persoanelor închiriază biciclete în jurul intervalelor orare 7 AM-8AM și 5 PM-6 PM. Se poate deduce că aceste interval reprezintă momentul începerii zilei de muncă/școală, respectiv sfârșitul acesteia.

- În zilele de weekend, intervalul orar de închiriere se modifică. (10AM-4PM)

- Un maxim de închirieri se distinge în jurul intervalului orar de 7 AM-8AM și 5 PM-6 PM, închirieri realizate de clienții abonați.

➢ *Test.csv*

După antrenarea modelului cu ajutorul *Random Forest* folosind ca input *train.csv*, modelul se testează cu ajutorul setului de date furnizat tot de Kaggle, *test.csv*.

Structura *test.csv* este următoarea:

```
datetime,season,holiday,workingday,weather,temp,atemp,humidity,windspeed
2011-01-20 00:00:00,1,0,1,1,10.66,11.365,56,26.0027
2011-01-20 01:00:00,1,0,1,1,10.66,13.635,56,0
2011-01-20 02:00:00,1,0,1,1,10.66,13.635,56,0
2011-01-20 03:00:00,1,0,1,1,10.66,12.88,56,11.0014
2011-01-20 04:00:00,1,0,1,1,10.66,12.88,56,11.0014
2011-01-20 05:00:00,1,0,1,1,9.84,11.365,60,15.0013
2011-01-20 06:00:00,1,0,1,1,9.02,10.605,60,15.0013
2011-01-20 07:00:00,1,0,1,1,9.02,10.605,55,15.0013
2011-01-20 08:00:00,1,0,1,1,9.02,10.605,55,19.0012
```

**Figura 27 - Mostră din *test.csv***

Se remarcă lipsa ultimelor 3 atribute, de aceea, pentru realizarea predicției, la implementarea aplicației setul de date de test se reformatează.

- **Antrenare model & testare**

Cu ajutorul algoritmului ales anterior, Random Forest, începe procesul de învățare. După ce "învață" din experiența anterioară ( input *train.csv*), modelul va fi testat și apoi validat de cei de la Kaggle pe baza RMSLE. Odată validat, va putea fi introdus în interfața pentru utilizator pentru a realiza predicțiile.

## 3.3 Design-ul aplicației

Aplicația se prezintă sub forma client-server. Partea de client conține două interfețe dedicate: una pentru *data analist* și una pentru utilizatorul/proprietarul afacerii. Partea de Machine Learning se va realiza în *background*, pe server (*Tomcat*) unde va fi efectuată predicția cu ajutorul *Random Forest* din *Weka* și a claselor din *Java*.

### 3.3.1 Viziune de ansamblu

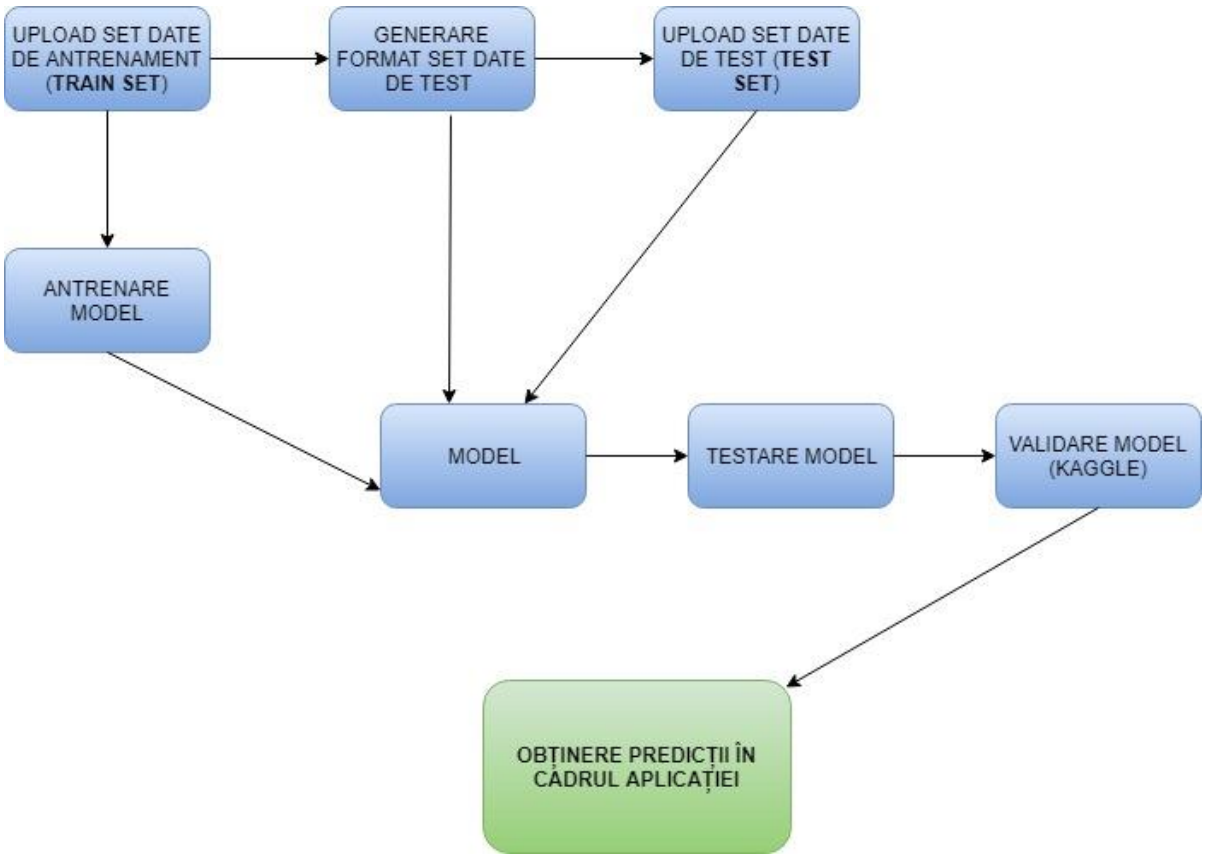


**Figura 28 - Structura de tip *pipeline***

*Workflow*-ul este cel caracteristic *task*-urilor de *Machine Learning*, ajustările/particularizările fiind realizate în etapa de preprocesare a datelor sau în timpul antrenamentului pentru a se obține un model cât mai eficient posibil.

Succesiunea generală a etapelor parcurse în logica aplicației este următoarea:

- Încărcare date de antrenament (*train set*) în format *.csv/.arff*
- Generare format corespunzător pentru datele de test (*test set*)

Inițial, setul de date de test furnizat de Kaggle nu era complet din punct de vedere al numărului de atribute. Din această cauză a fost necesară construirea "oglinditului" setului de date de antrenament, înlocuind atributele lipsă cu "?". Nepotrivirea celor două seturi de date (*train set* și *test set*) ar fi condus la imposibilitatea de testare a modelului, implicit de lansarea ulterioară a acestuia în "producția" de predicții.

- Antrenare model utilizând algoritmul Random Forest din librăria Weka
- Generare fișier de *output* cu predicțiile obținute
- Descărcare fișier output
- Evaluare model pe platforma Kaggle (se uploadează fișierul de *output* obținut, Kaggle evaluează pe baza RMSLE, acordă un scor, urmând ca soluția să fie inclusă în clasamentul lor general.)
- Vizualizare scor obținut în interfața dedicată *data analist*-ului
- Realizare predicții pentru o anumită instanță în interfața dedicată *user*-ului
- Vizualizare predicție

### 3.3.2 Diagrama use case

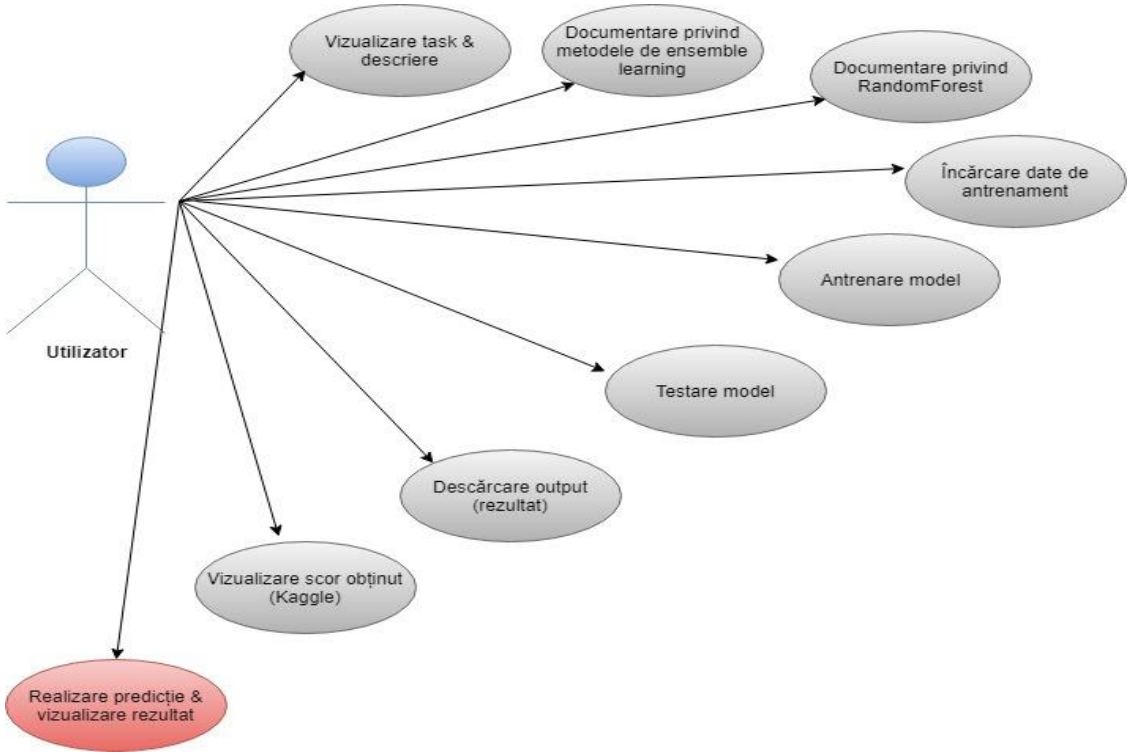


**Figura 29 - Diagrama use case**

### 3.3.3 Ierarhia de clase & metode Java

- *AddAttribute.java*
  - *createTestFormat()* – generează formatul adecvat al fișierului de test (*reverse train set*)

- *DataModel.java*
  - *get & set* pentru fiecare parametru (atribut) în parte – util pentru validarea câmpurilor din interfața pentru user (obținere predicție)

- *TestClassifiers.java*
  - *validate( )* – validare câmpurile după completare pentru a realiza predicția
  - *creare listă* cu parametrii (atributele corespunzătoare)
  - *inst_co.setValue* – permite adăugarea din interfață a unei instanțe la fiecare atribut pentru a realiza predicția
  - *BufferedReader reader* – citire input (*train set*) conform *filepath*-ului

- *RandomForest rf. buildClassifier* – apel Random Forest din librăria Weka & antrenare model
- *eval.crossValidateModel* – evaluare model
- *test.setClassIndex* – testare model
- *saver.setFile* – salvare fișier cu *outputul* obținut pe *hard-disk*

- *WebContent*
  - *dataanalistinterface.jsp* – interfața pentru *data analist*
  - *userinterface.jsp* – interfața pentru utilizator/proprietar
  - *userinterface2.*jsp – utilizat pentru validarea unei instanțe în vederea predicției
  - *popup.jsp* – *popup* cu predicția
  - *uploadtrain.jsp* – încărcare date de antrenament
  - *downloadoutput.jsp* – descărcare output

### 3.3.4 Interfața cu utilizatorul

Din punct de vedere al interfeței, aplicația are o dublă perspectivă:

- Interfața pentru *data analist*, cu următoarele funcționalități:
  - Vizualizare descriere problemă și cerință
  - Acces la documentație privind *ensemble learning*
  - Acces la documentație privind *RandomForest*
  - Upload *train set*
  - Antrenare & testare model (+rulare *TestClassifiers* din Eclipse)
  - Download *output file*
  - Vizualizare scor după evaluare pe platforma Kaggle

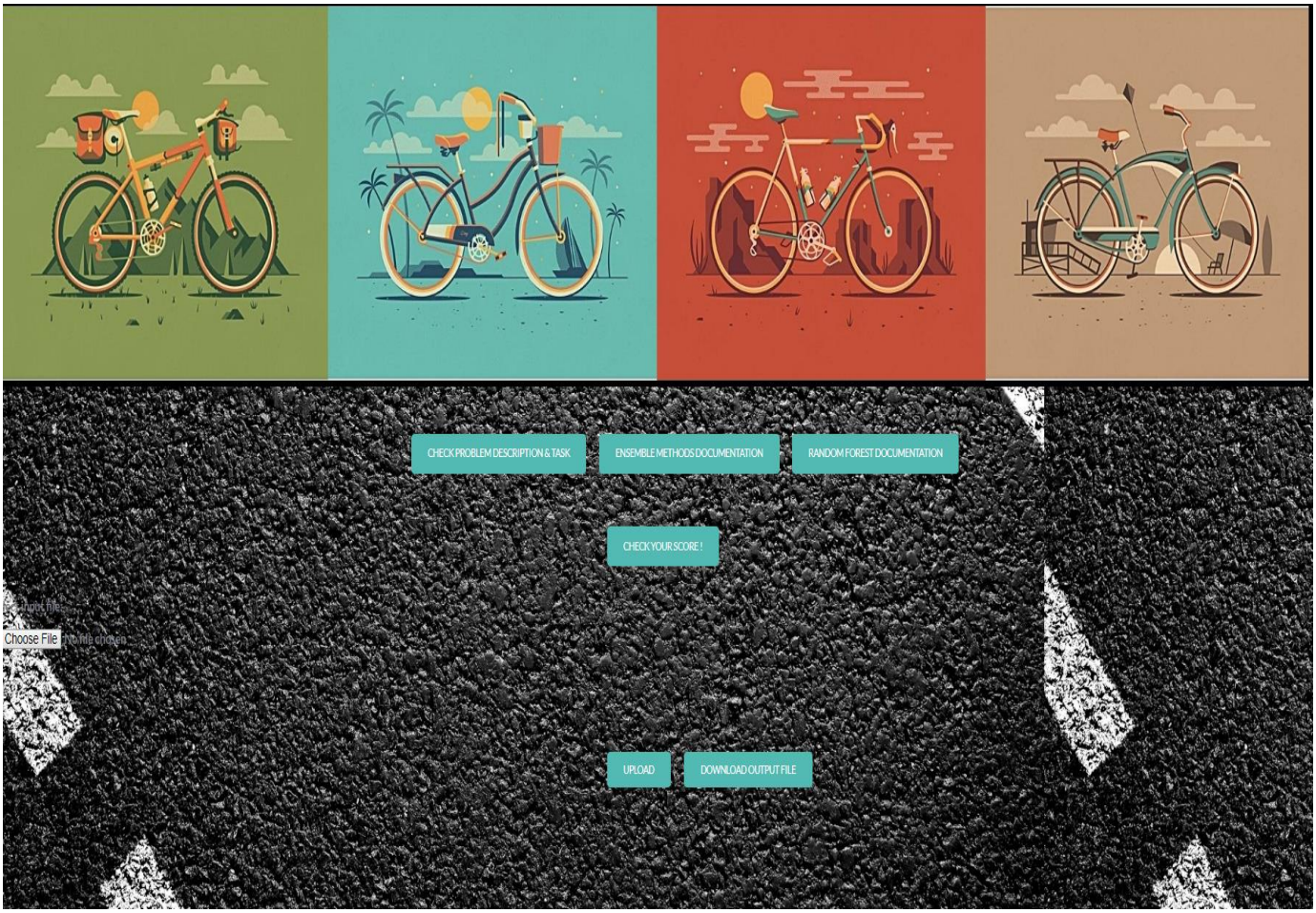


**Figura 30 - Interfața pentru *data analist***

- Interfața pentru utilizator/proprietar cu următoarele funcționalități:
  - Introducere parametri (atribute) pentru o instanță în vederea obținerii predicției
  - Vizualizare predicție
  - Caracterizată de noțiunea de *black box vis-à-vis* de cod, pentru a păstra aspectul de *user-friendly*. Logica *Machine Learning* rămâne însă în *background*.

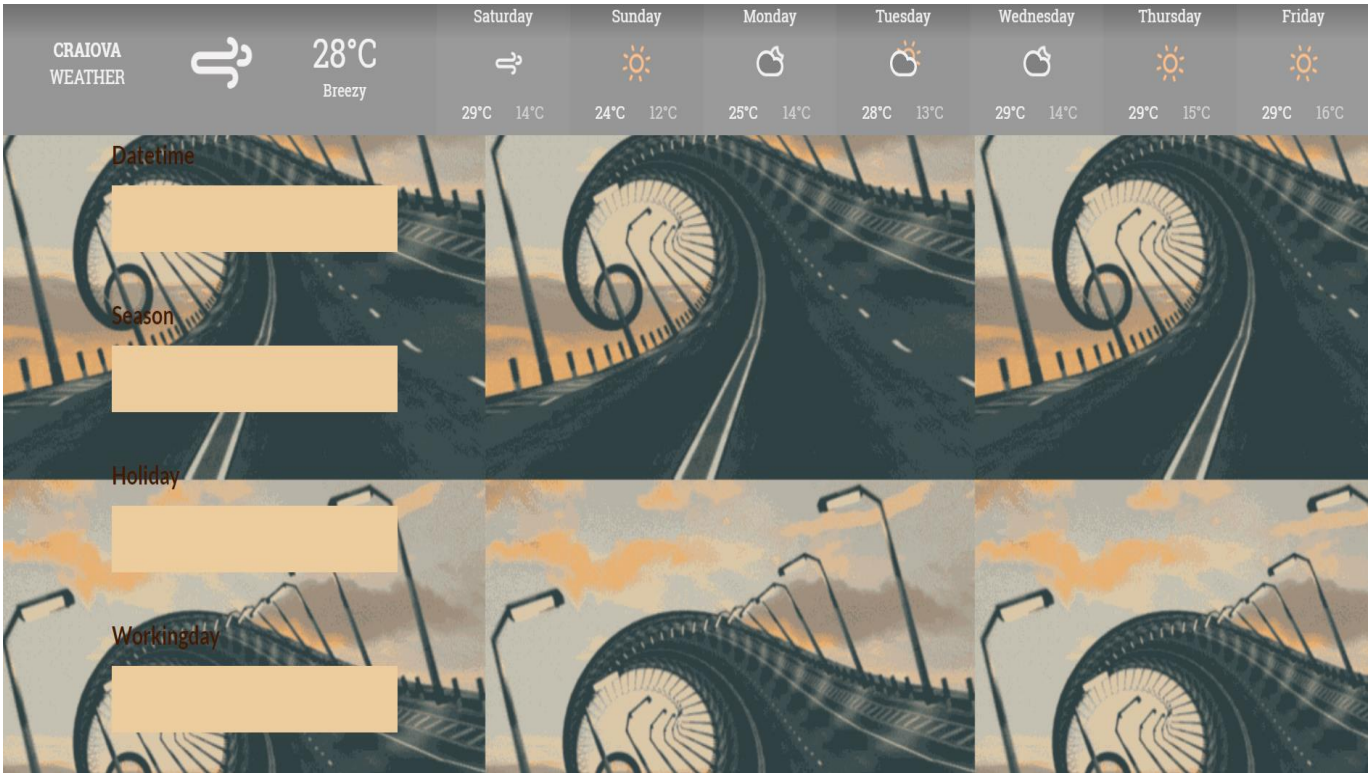


**Figura 31 - Interfața pentru utilizator**

### 3.3.5 Evaluarea output-ului

După antrenarea, validarea și testarea modelului în cadrul realizării aplicației, evaluarea oficială a fost realizată de cei de la Kaggle. Fișierul cu rezultate obținute (*output*) generat în format *.csv* a fost încărcat la Kaggle și evaluat, acordându-i-se un scor. Acest scor a permis plasarea soluției proprii în clasamentul oficial *worldwide*.

Conform evaluării celor de la Kaggle, cu RMSLE și diferite seturi de date proprii, scorul obținut prin soluția propusă (aplicația curentă) este de **1.35034**.

Plaja de valori în care au fost date soluții la problemă până la momentul propunerii soluției proprii a fost **0.34 – 4.77** (0.34 fiind considerată cea mai bună soluție propusă), ceea ce plasează această soluție în zona de "model eficient, util în producție", putând fi utilizat pentru realizarea predicțiilor cu marja de eroare aferentă.

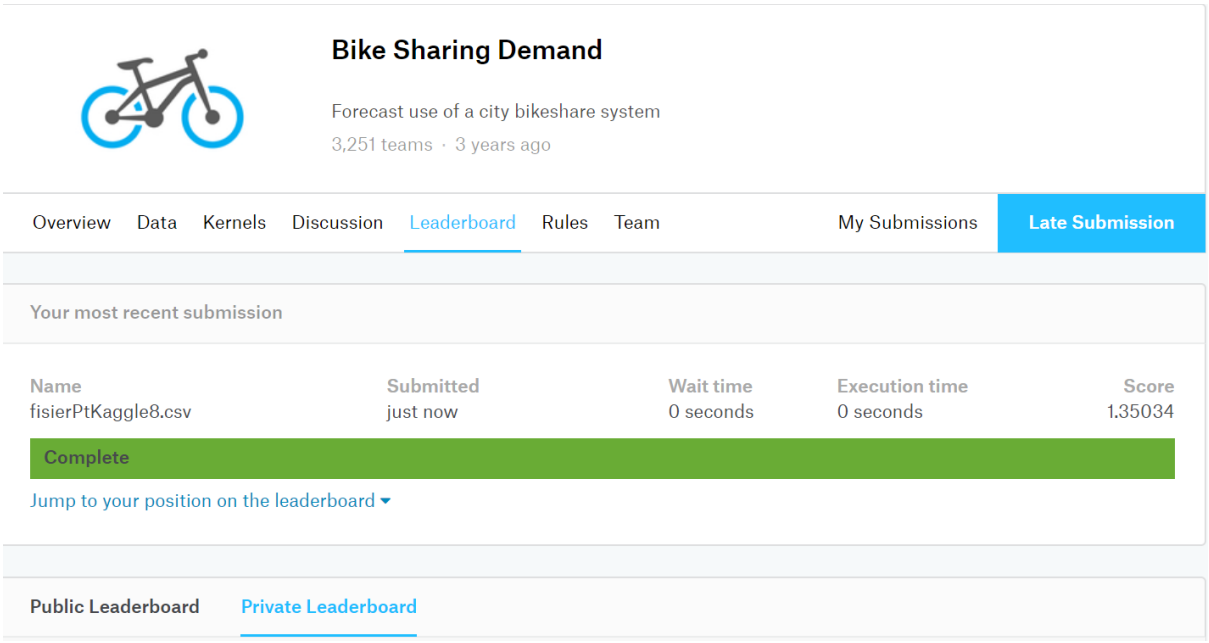


**Figura 32 - Scor obținut cu soluția proprie**

Comparativ, pentru producerea unei soluții cu un set de date de antrenament redus pentru un singur anotimp (primăvara), scorul obținut a fost unul ușor îmbunătățit, **1.34855**, ceea ce indică o creștere a acurateții odată cu reducerea setului de date de antrenament.

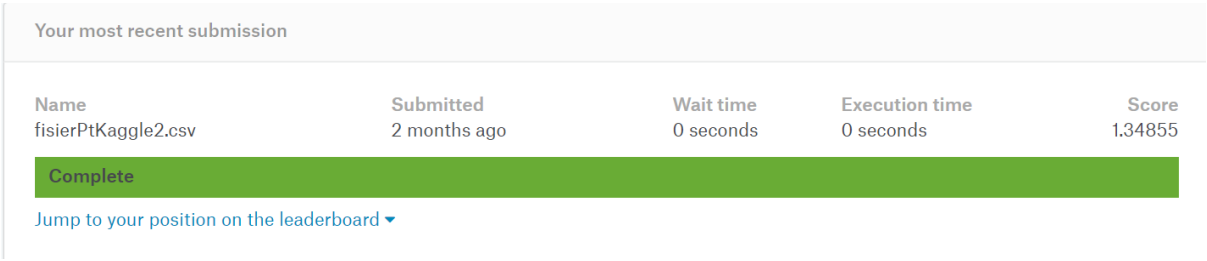


**Figura 33 - Scor obținut cu setul de date restrâns la un anotimp**

# 4 CONCLUZII

La momentul actual, domeniul *Machine Learning* este în plină expansiune, am putea spune chiar "tehnologia momentului", ceea ce americanii ar denumi pur și simplu "*buzzword*". Odată cu evoluția tehnologiei și lărgirea orizontului academic al inginerilor, *Machine Learning*-ul a devenit indispensabil, cel în jurul căruia gravitează toate inovațiile tehnologice. Toate acestea au constituit un motiv solid pentru mine în alegerea abordării acestei teme. Ideea de a dezvolta o aplicație/un proiect care să aibă aplicabilitate în viața reală și care să fie simultan și "extrem de actuală, în trend" a devenit o adevărată provocare, mai ales că soluția pe care urma să o propun beneficia de o evaluare oficială, a celor de la Kaggle.

Kaggle reprezintă o platformă recunoscută la nivel mondial în domeniul *data science,* care prin competițiile găzduite a furnizat soluții la diverse probleme ce implică algoritmi de Machine Learning și predicții, de la îmbunătățirea sistemului *Microsoft Kinect,* cercetarea privind *bosonul Higgs* ("*particula Lui Dumnezeu"*), diverse studii în domeniul medical (patologii de boli, evaluare predispoziție la diferite afecțiuni), strategii de marketing, șah, piața imobiliară, studiul mobilității populației (unde se încadrează și această aplicație), până la *Watson* dezvoltat de IBM, care a fost considerat o revelație în domeniul *Artificial Intelligence*.

Aplicația realizată este în primul rând utilă vieții cotidiene, fiind un instrument de suport în alegerea strategiilor de marketing potrivite acestui tip de afacere, acel "secret" care îl poate ajuta pe proprietar să fie mereu cu un pas înaintea concurenței, să anticipeze. Pe lângă utilitate și aplicabilitate în viața reală, aplicația este una flexibilă, putând fi customizată și pentru alt tip de afacere. În fond, ideea centrală o constituie realizarea predicțiilor, particularitatea fiind dată de alegerea atributelor (parametrilor) și colectarea seturilor de date aferente.

Pe parcursul dezvoltării acestei aplicații, consider că mi-am îmbunătățit abilitățile de programare și proiectare a unei soluții. De asemenea, mi-am consolidat cunoștințele în domeniul *Machine Learning,* pe care l-am considerat extrem de interesant și aparte în ceea ce privește ingineria datorită conceptului de *Artificial Intelligence* care prin definiție este unul fascinant.

Realizarea aplicației s-a concretizat în studiul și utilizarea algoritmului *Random Forest,* care mi-a atras atenția în mod special în urma lecturii unui articol în care este dezvoltată ideea că acest algoritm este unul extrem de flexibil și care prin diverse modificări/ajustări/îmbunătățiri poate oferi rezultate remarcabil superioare celor obținute prin metoda clasică.

“[…] În plus, încercăm să utilizăm diferiți algoritmi de *Random Forest* pentru a rezolva problema oceanografică extrem de provocatoare, aceea a recunoașterii fronturilor oceanice. Din cunoștințele noastre, este prima dată când folosim *Random Forest* pe o problemă de acest tip. Un **front oceanic** este o limită între două mase distincte de apă. Masele de apă sunt definite prin deplasarea în direcții diferite, adică, pe o parte a frontului oceanului, apa se mișcă, în general, într-un sens, iar pe cealaltă parte a frontului oceanului, apa se mișcă în alt sens. Masele de apă de pe ambele părți ale frontului oceanic pot avea, de asemenea, temperaturi diferite, salinități sau densități, împreună cu diferențe în cazul altor marcatori oceanografici. Frontul oceanului are o semnificație importantă în aplicații în domeniul pescuitului marin, al protecției mediului, al afacerilor militare marine etc. **Recunoașterea** și **detectarea frontului oceanic** au devenit subiectele importante de studiu în Oceanografie fizice și disciplinei marine. Conform rezultatului experimental al recunoașterii frontului oceanic, acesta a arătat că CPRF (*Cooperative Profit Random Forests*) a depășit performanțele altor algoritmi de *Random Forests* deja existenți.

CPRF folosește o nouă metodă de împărțire a nodurilor pentru a dezvolta cooperarea între nodurile de arbori ale clasificatorului individual de în cadrul *Random Forests* bazate pe ideea teoriei cooperative a jocurilor. (*cooperative game theory*)

Odată cu îmbunătățirea treptată a rezoluției spațiale a teledetecției (RS), datele de înaltă rezoluție prin satelit sunt din ce în ce mai accesibile și mai ieftine. Din datele furnizate de sateliți (imagini de înaltă rezoluție), putem avea acces de la distanță la informații detaliate referitoare la aranjarea spațială și structurile texturale. Fronturile oceanice au atras un mare interes de cercetători. Majoritatea cercetătorilor se concentrează doar pe problema **detectării frontului oceanic**. Cele mai populare metode includ algoritmi de gradient, algoritmi de entropie. Cu toate acestea, metodele menționate au o eficiență mai mică pentru detectarea frontului oceanului. Vom folosi algoritmi diferiți de *Random Forest* pentru a clasifica regiunile. […]” (https://ieeexplore.ieee.org/document/7828092/ )

În concluzie, aplicația este utilă, flexibilă și constituie un element de orientare în planul strategic al utilizatorului/proprietarului afacerii în zona de *business/marketing*, putând însă căpăta și o dublă perspectivă în ceea ce privește interesul cercetătorilor pentru mobilitatea populație dintr-o anumită zonă. În plus, algoritmul utilizat (*Random Forest*) are proprietatea de a dobândi valențe superioare în ceea ce privește eficiența sa, ceea ce consolidează posibilitatea de îmbunătățire în viitor a aplicației dezvoltate pentru acest proiect de diplomă.

# 5 BIBLIOGRAFIE

*[AS08] INTRODUCTION TO MACHINE LEARNING, editura Cambridge University Press 2008, disponibilă la adresa http://alex.smola.org/drafts/thebook.pdf*

*[IG16] DEEP LEARNING, MIT 2016, disponibilă la adresa https://www.deeplearningbook.org/front_matter.pdf*

*[TM97] MACHINE LEARNING, editura McGraw-Hill Science/Engineering/Math 1997, disponibilă la adresa https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf*

# 6 REFERINȚE WEB

*[1] Machine Learning. What it is and why it matters, disponibil la adresa https://www.sas.com/en_us/insights/analytics/machine-learning.html*

*[2] Machine Learning, disponibil la adresa https://en.wikipedia.org/wiki/Machine_learning*

*[3] What Is The Difference Between Artificial Intelligence And Machine Learning ?, disponibil la adresa https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-machine-learning/#71efc0602742*

*[4] 10 Real-World Examples of Machine Learning and AI [2018], disponibil la adresa https://www.redpixie.com/blog/examples-of-machine-learning*

*[5] What Is The Difference Between Artificial Intelligence And Machine Learning ?, disponibil la adresa https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-machine-learning/#6b6404622742*

*[6] The Random Forest Algorithm, disponibil la adresa https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd*

*[7] Java programming language, disponibil la adresa https://en.wikipedia.org/wiki/Java_(programming_language)*

*[8] Despre mașina virtuală Java, disponibil la adresa http://control.aut.utcluj.ro/isp/lab1/DespremasinavirtualaJava.htm*

*[9] Apache Tomcat, disponibil la adresa https://en.wikipedia.org/wiki/Apache_Tomcat*

*[10] Weka software, disponibil la adresa https://en.wikipedia.org/wiki/Weka_(machine_learning)*

*[11] How to get started with Kaggle, disponibil la adresa https://machinelearningmastery.com/get-started-with-kaggle/*

*[12] Random Forest, disponibil la adresa https://en.wikipedia.org/wiki/Random_forest#Preliminaries:_decision_tree_learning*

*[13] A Comprehensive Guide to Ensemble Learning, disponibil la adresa https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/*

*[14] Tuning the parameters of your Random Forest model, disponibil la adresa https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/*

*[15] Random Forests, disponibil la adresa http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/RandomForests.pdf*

*[16] Machine Learning with Java, disponibil la adresa https://tech.io/playgrounds/7163/machine-learning-with-java---part-6-random-forest*

*[17] Cooperative Profit Random Forests With Application in Ocean Front Recognition, disponibil la adresa https://ieeexplore.ieee.org/document/7828092/*

*[18] Kaggle competitions, disponibil la adresa https://medium.com/@viveksrinivasan/how-to-finish-top-10-percentile-in-bike-sharing-demand-competition-in-kaggle-part-1-c816ea9c51e1*

*[19] Bike Sharing Demand – Kaggle, disponibil la adresa https://www.kaggle.com/c/bike-sharing-demand*

*[20] HOW TO CALCULATE PERCENTILES IN STATISTICS, disponibil la adresa https://www.dummies.com/education/math/statistics/how-to-calculate-percentiles-in-statistics/*

*[21] JSP Tutorial, disponibil la adresa https://www.javatpoint.com/jsp-tutorial*

*[22] Weka programmatic use, disponibil la adresa http://weka.wikispaces.com/Programmatic+Use*

*[23] A Simple Machine Learning Example in Java, disponibil la adresa https://www.programcreek.com/2013/01/a-simple-machine-learning-example-in-java/*

# CODUL SURSĂ

# CD/DVD

# INDEX