нују се и процес и продукти учења. Прикупљање информација из различитих извора (свакодневна посматрања, активност на часу, учествовање у разговору и дискусији, самосталан рад, рад у групи, тестови) помаже наставнику да сагледа постигнућа (развој и напредовање) ученика и степен остварености исхода. Свака активност је добра прилика за процену напредовања и давање повратне информације. Важно је и ученике оспособљавати и охрабривати да процењују сопствени напредак у учењу.

У процесу праћења и вредновања значајну улогу имају домаћи задаци. Редовно задавање домаћих задатака (уз обавезну повремену проверу од стране наставника), помаже наставнику да стекне бољи увид у степен остварености исхода кроз анализу задатака које ученици нису умели да реше. Важно је и мотивисати ученике који редовно раде домаће задатке тако што ће њихов рад бити оцењен.

Вредновање активности у оквиру тимског рада се може обавити са групом тако да се од сваког члана тражи објашњење елемената урађеног рада и мишљење о сопственом раду унутар тима. Препоручује се да наставник са ученицима договори показатеље на основу којих сви могу да прате напредак у учењу, ученици се уче да размишљају о квалитету свог рада и о томе шта треба да предузму да би свој рад унапредили. Оцењивање тако постаје инструмент за напредовање у учењу. На основу резултата праћења и вредновања, заједно са ученицима треба планирати процес учења и бирати погодне стратегије учења.

Препоручено је да коначна оцена за сваког ученика буде добијена комбиновањем различитих начина оцењивања:

– активност на часу, учествовање у разговору и дискусији;
– редовна израда домаћих задатака;
– писмени задаци;
– тестови – провера знања;
– пројектни рад, и појединачни и тимски.

Комбиновање различитих начина оцењивања помаже да се сагледају слабе и јаке стране сваког ученика. Приликом сваког вредновања постигнућа потребно је ученику дати повратну информацију која помаже да разуме грешке и побољша свој резултат и учење. Потребно је да наставник резултате вредновања постигнућа својих ученика континуирано анализира и користи тако да промени део своје наставне праксе.

ПРОГРАМСКЕ ПАРАДИГМЕ

Циљ учења предмета Програмске парадигме је да ученици, кроз упознавање са програмским парадигмама и вештачком интелигенцијом, развију компетенције за програмирање и одговорно коришћење информационо-комуникационих технологија, као и да ученике оспособи за примену усвојених знања из области рачунарства и информатике, решавање разноврсних проблема из животне праксе, да их припреми за наставак образовања, да допринесе развијању менталних способности, формирању научног погледа на свет и свестраном развитку личности ученика.

ОПШТА ПРЕДМЕТНА КОМПЕТЕНЦИЈА

Учењем наставног предмета Програмске парадигме ученици су оспособљен да примене стечена знања и вештине из области информационо-комуникационих технологија ради испуњавања постављених циљева и задатака у свакодневном животу, даљем школовању и будућем раду. Ученици развијају способност апстрактног и критичног мишљења уз помоћ информационо-комуникационих технологија. Развијају дигиталну писменост и позитивне ставове према рачунарским наукама.

СПЕЦИФИЧНЕ ПРЕДМЕТНЕ КОМПЕТЕНЦИЈЕ

Специфичне предметне компетенције представљају опис специфичних способности ученика које му омогућавају да развије општу предметну компетенцију. Ученици одговорно користе информационо-комуникационе технологије уз препознавање потенцијалних ризика и опасности. Решавају практичне проблеме применом различитих програмских парадигми (логичко програмирање, функционално програмирање и машинско учење као парадигма програмирања заснована на подацима).

Разред	**Четврти**
Недељни фонд часова	**3 часа вежби**
Годишњи фонд часова	**99 часова вежби**

ИСХОДИ По завршетку разреда ученик ће бити у стању да:	**ТЕМА** и кључни појмови садржаја програма
– опише основне карактеристике најпознатијих програмских парадигми – класификује програмске језике на основу програмских парадигми – разликује и кроз примере илуструје декларативно и императивно програмирање – објасни логичке основе логичког програмирања; – објасни појам ваљане формуле логике првог реда; – опише испитивање ваљаности применом метода резолуције на клаузалну форму негације полазне формуле; – дефинише једноставну базу знања и правила закључивања; – на основу постављених циљева добије информације из базе знања; – објасни процес израчунавања одговора коришћењем стабла израчунавања одговора; – примени рекурзију у раду са листама; – опише дејство реза (црвеног и зеленог) и примењује га у решавању проблема; – реши комбинаторне проблеме и логичке загонетке применом логичког програмирања. – наведе основне карактеристике функционалне парадигме; – наведе мане употребе споредних (бочних) ефеката у програмирању и начине њиховог избегавања; – наведе примере разлагања проблема на једноставније потпроблеме и њиховог решавања композицијом функција; – дефинише изразе и функције засноване на изразима; – примени основне функције и функционале вишег реда над листама; – дефинише функције коришћењем рекурзије; – дефинише корисничке (алгебарске) типове података, укључујући и генеричке типове и функције које их обрађују; – препозна и употреби елементе функционалног програмирања у савременим мултипарадигматским језицима. – препозна употребу машинског учења у својој околини; – кроз примере илуструје парадигму програмирања заснованог на подацима; – разликује и кроз примере илуструје класе проблема, метода и модела машинског учења; – на основу дате примене предложи метод машинског учења и објашњава поступак прикупљања података;	**УВОД У ПРОГРАМСКЕ ПАРАДИГМЕ** – Појам програмске парадигме – Декларативно и императивно програмирање – Класификација програмских језика на основу парадигме којој припадају – Комбиновање програмских парадигми у једном програмском језику – Процедурална парадигма – Објектнооријентисана парадигма – Скрипт парадигма **ЛОГИЧКО ПРОГРАМИРАЊЕ** – Основне карактеристике логичког програмирања. – Логичке основе логичког програмирања (предикатска логика првог реда, Хорнове клаузуле, унификација и резолуција) – Основни елемената синтаксе изабраног логичког програмског језика (неке варијанте језика PROLOG): константе, променљиве, терми – Програмске клаузуле (чињенице, правила и циљеви) – Принцип израчунавања одговора (унификација, бектрекинг), стабло израчунавања – Дефинисање функција у облику релација – Рекурзија – Аритметичка израчунавања (оператор is) – Листе – Рез и примена реза, врсте реза (црвени и зелени рез) – Решавање комбинаторних проблема и логичких загонетки

– примени готов модел машинског учења за решавање задатог проблема; – реши једноставан класификациони проблем на основу прикупљених података; – разуме и интерпретира мерења квалитета система машинског учења; – објашњава практичне и етичке проблеме употребе машинског учења;	**ФУНКЦИОНАЛНО ПРОГРАМИРАЊЕ** – Основне карактеристике функционалне парадигме (одсуство споредних ефеката, референцијална транспарентност, композиционалност, имутабилност, лењо израчунавање, ...) – Примери решавања проблема коришћењем функционалног начина размишљања – Основни елементи изабраног програмског језика. – Систем типова (основни типови, функцијски типови, Каријеве функције) – Изрази (константе, оператори, if-then-else, let-in, where, ламбда изрази, ...) – Дефинисање функција коришћењем израза, упаривање шаблона – Листе и рад са листама – Функције вишег реда (map, filter, fold) – Дефинисање рекурзивних функција – Кориснички типови (алгебарски типови, рекурзивни типови, генерички типови, параметарски полиморфизам, ...) – Проблеми коришћења чистих функционалних језика (нпр. улаз-излаз, изоловање споредних ефеката, функтори, монаде, ...)
	САВРЕМЕНА ВЕШТАЧКА ИНТЕЛИГЕНЦИЈА – Појам вештачке интелигенције и машинског учења. Укључујући историју и узроке пробоја савремених приступа – Преглед примера употребе машинског учења – Машинско учење као парадигма програмирања заснована на подацима. Примери синтезе програма коришћењем машинског учења – Појам прикупљања, организације и лабелирања података неопходних за машинско учење – Класификација проблема, метода и модела машинског учења (регресија, кластеризација, класификација, супервизијско, несупервизијско, перцептрон, неуралнa мрежа, …) – Појам класичног машинског учења насупрот дубоком учењу. Интуитивно разумевање дубоког учења – Употреба готовог модела машинског учења у практичним примерима (прављење апликације за детекцију лица, апликације за класификацију текста…) – Мерење квалитета модела машинског учења (тачност и поновљивост, матрица конфузије, лажни позитиви, негативи…) – Решавање класификационог проблема на основу модела k најближих суседа – Практични проблеми и ограничења употребе машинског учења – Етички проблеми употребе машинског учења

УПУТСТВО ЗА ДИДАКТИЧКО-МЕТОДИЧКО ОСТВАРИВАЊЕ ПРОГРАМА

Настава вежби се изводи са половином одељења у рачунарском кабинету, у групама не већим од 12 ученика.

I. ПЛАНИРАЊЕ НАСТАВЕ И УЧЕЊА

Приликом планирања часа, исходе предвиђене програмом треба разложити на мање и на основу њих планирати активности за конкретан час. Треба имати у виду да се исходи у програму разликују, да се неки могу лакше и брже остварити, док је за одређене исходе потребно више времена, активности и рада на различитим садржајима. Исходе треба посматрати као циљеве којима се тежи током једне школске године.

При обради нових садржаја треба се ослањати на постојеће искуство и знање ученика, и настојати, где год је то могуће, да ученици самостално откривају математичке правилности и изводе закључке. Ученике треба упућивати да користе уџбеник и друге изворе знања, како би усвојена знања била трајнија и шира, а ученици оспособљени за примену у решавању разноврсних задатака.

На часовима треба комбиновати различите методе и облике рада, што доприноси већој рационализацији наставног процеса, подстиче интелектуалну активност ученика и наставу чини интересантнијом и ефикаснијом. Препоручује се коришћење интерактивних метода, пројектне, проблемске и истраживачке методе, дискусије, дебате и др, како би ученици били што више ангажованом током наставе. Комбиновати на часовима различите облике рада као што су самостални рад ученика (по принципу један ученик – један рачунар), рад у паровима (два ученика истовремено и заједно решавају конкретне задатке), рад у мањим групама (почетна анализа и идеје за методе решавања), као и рад са целом групом када наставник објашњава, приказује, демонстрира и кроз дискусију уводи ученике у нове области. Избор метода и облика рада, као и планирање активности ученика ускладити са наставним садржајем који треба реализовати на часу и предвиђеним исходима, али и са специфичностима одељења и индивидуалним карактеристикама ученика.

Предложени број часова по темама је оквирни, на наставнику је да процени потребан и довољан број часова по темама узимајући у обзир знања и вештине који ученици имају из претходног школовања и животног искуства. Предложени редослед тема није обавезујући за наставнике, већ само представља један од могућих модела, који наставник може прилагодити у складу са изабраним програмским језиком и методолошким опредељењем.

Ради лакшег планирања наставе даје се оријентациони предлог броја часова по темама:

– Преглед програмских парадигми (5 часова)
– Логичко програмирање (26 часова)
– Функционално програмирање (26 часова)
– Савремена вештачка интелигенција (42 часа)

II. ОСТВАРИВАЊЕ НАСТАВЕ И УЧЕЊА

У оквиру теме **Преглед програмских парадигми** потребно је:

Упознати ученике са основним појмом програмских парадигми и дати кратак преглед историјата развоја и класификације програмских језика. Детаљнији преглед парадигми започети подсећањем ученика на основне карактеристике процедуралне парадигме која им је позната из ранијих разреда. Истаћи разлике процедуралне и објектно-оријентисане парадигме и скренути пажњу ученицима на то да су се они у ранијем школовању заправо већ сусрели са различитим програмским парадигмама. Кроз дискусију упоредити однос те две парадигме и продискутовати искуство ученика у њиховом коришћењу. Упознати ученике са односом императивног и декларативног програмирања, са особинама декларативног програмирања и начином описивања проблема у декларативним програмским језицима. Скренути пажњу ученицима да велики број савремених програмских језика комбинује елементе више парадигми. Истаћи програмирање засновано на подацима и машинско учење као посебну парадигму која се у савременом рачунарству користи све интензивније. Истакнути аутоматску синтезу кода, продискутовати њене тренутне могућности и импликације на процес програмирања у будућности.

У оквиру теме **Логичко програмирање** потребно је ученике упознати са основним карактеристикама логичке парадигме и

истаћи да се ова парадигма темељи на логици првог реда. Укратко описати историјат развоја логичке парадигме и улогу логичког програмирања и аутоматског резоновања у традиционалним системима вештачке интелигенције.

Логика се користи као декларативни језик за опис проблема, а доказивач теорема уграђен у програмски језик за решавање проблема. Истаћи да у логичком програмирању програмер проблем описује као скуп логичких формула (односа), а систем аутоматски решава проблем извођењем одговарајућих логичких закључака. У циљу бољег разумевања карактеристика логичког програмирања ученицима се већ на првом часу може приказати једноставан логички програм (на пример, база знања и скуп правила закључивања за анализу породичних односа).

Са ученицима је потребно обновити градиво из исказне логике, па затим упознати ученике са основама логике првог реда. Дефинисати синтаксу предикатских формула (језик као скуп релацијских и функцијских симбола, термове, атомичке формуле и на крају формуле). Дефинисати затим и семантику и описати како се одређује тачност формуле када се фиксирају домен и интерпретација симбола. Дефинисати појам ваљане формуле (формуле која је тачна при свим интерпретацијама). Објаснити да се испитивање ваљаности најчешће врши методом резолуције (испитивањем да је негација формуле незадовољива), а да одређени облик метода резолуције представља основу логичког програмирања и програмског језика PROLOG. Приказати да се метода резолуције примењује на формуле у клазуалној форми (увести појам клаузуле и литерала). Описати поступак превођења произвољне формуле у клаузалну форму (описати процес трансформације произвољне предикатске формуле у еквивалентну пренекс нормалну форму, процес сколемизације и процес превођења у конјунктивну нормалну форму). Описати ученицима проблем унификације два израза и процес налажења најопштијег унификатора.

Објаснити метод резолуције логике првог реда и примена метода резолуције при испитивању да ли је скуп клаузула незадовољив. У циљу лакшег разумевања метода резолуције логике првог реда, може се објаснити прво метод резолуције исказне логике и његова примена.

Упознати ученике са Хорновим клаузулама, клазулама у којима постоји највише један литерал који је под негацијом. Указати на чињеницу да Хорнове клаузуле омогућавају ефикасну примену метода резолуције. Истаћи да је програмски језик PROLOG заснован је на методу резолуције и коришћењу Хорнових клаузула. На примеру програма за рад са породичним стаблом приказати везу између PROLOG-а и Хорнових клаузула и метода резолуције.

Напомена: У зависности од интересовања ученика и расподеле осталих часова, могуће је са ученицима детаљније обрадити логичке основе логичког програмирања.

Упознати ученике са синтаксом програмског језика PROLOG:

– упознати ученике са појмом терма (константе, променљиве, структуре) као основним градивним елементом у PROLOG-у;

– упознати ученике са различитим врстама програмских клаузула (чињенице, правила и циљеви);

– упознати ученике са процесом унификације у PROLOG-у;

– упознати ученике са процесом израчунавање одговора; објаснити начин креирања стабла израчунавања свих одговора за дати циљ, као и обилазак стабла који PROLOG ради претрагом по дубини (претрага са враћањем – бектрекинг), на неколико примера приказати процес израчунавања одговора.

Нагласити да се програмирање у PROLOG-у састоји од записивања чињеница о објектима и односима између објектима, дефинисању правила о објектима и односима међу њима, и формирању упита (циљева) о објектима и односима међу њима.

Упознати ученике са аритметичким и релацијским операторима у PROLOG-у, као и са системским предикатима *is* и *not*. При увођењу предиката *not* потребно је нагласити разлику између негације у PROLOG-у и логичке негације (у PROLOG-у циљ *not(C)* успева ако и само ако циљ *C* не успева). Нагласити и на примерима показати да је рекурзивно дефинисање релација темељни принцип програмирања у PROLOG-у. Нагласити да се функције не дефинишу директно, већ као релације код којих се непознати аргументи израчунавају на основу аргумената који су познати.

Дефинисати сложене структуре података, листе, као структуре разноврсних података са утврђеним редоследом, чијим елементима се приступа од првог елемента. Листа је један од кључних структура која се користи у PROLOG-у. Нагласити рекурзивну дефиницију листа и рекурзивни приступ решавању проблема са листама. Дефинисати основне предикате за рад са листама:

– број елемената листе,

– припадност елемента листи,

– спајање две листе,

– брисање елемента из листе,

– сортирање листе (различитим алгоритмима).

Показати како се у неким случајевима један предикат може користити за више функционалности за рад са листама у зависности од тога који аргумент тражимо (више функција се реализује једном релацијом тј. предикатом). На пример предикат *element(X, L)* којим се проверу да ли је *X* елемент листе *L* можемо користити и за издвајање свих елемената дате листе. На пример на питање *?– element(X,[1,7,2])* добијамо одговоре *X=1; X=7; X=2;*. Слично можемо показати да предикат којим се спајају две листе у трећу *spoji(L1, L2, L)* можемо користити за добијање свих листа *L1* и *L2* чијим спајањем добијамо трећу дату листу *L*.

Упознати ученике са оператором сечења – резом. Указати на разлику између црвеног и зеленог реза. Детаљно објаснити како рез функционише, и указати на примерима као је погрешна употреба реза чест узрок грешке у PROLOG-у, али и како исправна употреба реза је непоходна за добијање ефикасних решења. Илустровати ефекат реза на стабло израчунавања одговора.

Дефинисати предикате за решавање комбинаторних проблема:

– пермутације,

– варијације,

– комбинације.

Нагласити примену PROLOG-а у решавању логичких проблема, на пример са ученицима решити Ајнштајнов проблем кућа, проблем вук-коза-купус, проблем мисионари и људождери, распоређивање дама на шаховској табли, разне логичке загонетке и слично.

Напомена: У зависности од интересовања ученика и расподеле осталих часова, могуће је са ученицима обрадити уграђене предикате за улаз и излаз, предикате за рад са клаузулама, за рад са базом знања, дефинисање корисничких оператора и слично. Као и дефинисати појам експертског система и креирати једноставан експертски систем за препознавање различитих облика, животиња, предмета и слично.

У оквиру теме **Функционално програмирање** потребно је истаћи значај функционалне парадигме у савременом програмирању и утицај функционалне парадигме на развој савремених програмских језика. Функционалну парадигму је могуће илустровати или на неком чистом функционалном језику (нпр. Haskell, F#, Lisp, Scheme, Clojure, ...) или на неком мултипарадигматском језику који у значајној мери подржава функционалне концепте (нпр. C#, JavaScript, ...), а могућа је и комбинација ова два приступа.

Истаћи како имутабилност и недостатак глобалног стања програма омогућавају да се програми праве математичком композицијом функција чије вредности зависе искључиво од улаза који су им прослеђени (референцијална транспарентност). Истаћи значај овог стила у смањењу броја потенцијалних грешака, олакшаној анализи програма и последично обезбеђивању коректности софтвера. Истаћи и значај имутабилности у паралелном и конкурентном програмирању.

Истаћи значај израза у функционалним језицима као и одсуство традиционалних наредби које модификују стање програма. Упоредити if-then-else израз у функционалним језицима са if-then-else наредбом у императивним програмским језицима. Истакнути одсуство наредбе доделе, па самим тим и петљи и нагласити како се контрола тока остварује на друге начине (на пример, рекурзијом). Описати начин записа позива функција у одабраном језику (префиксни запис у језицима попут LISP-а, или Каријев запис у језицима попут Haskell-а).

Увести функције вишег реда које у комбинацији са анонимним функцијима омогућавају апстрактније и концизније изражавање алгоритама. Нарочито инсистирати на пресликавању (map), филтрирању (filter) и агрегирању тј. редуковању (reduce тј. fold) и посебним, најчешће коришћеним облицима редуковања (сумацији, бројању, проналажењу минимума и максимума, израчунавању производа, универзалној и егзистенцијалној квантификацији и слично). Посебну пажњу посветити концепту прослеђивања једне функције као аргумента другој. Истаћи употребу анонимних функција тј. ламбда израза у том контексту. Ако језик допушта Каријеве функције, приказати како се њиховом парцијалном инстанцијацијом могу на веома лак начин добити жељени параметри функција вишег реда (на пример, увећавање свих елемената листе xs у језику Haskell се може добити позивом map (\ x -> x + 1) xs, али и једноставнијим позивом map (+1) xs.

Истаћи значај листа које (нарочито у комбинацији са лењошћу) представљају облик организације контроле тока програма. Увести функције које генеришу листе (на пример, понављањем истог елемента, на основу неке правилности, попут аритметичких и геометријских низова, издвајањем цифара датог броја и слично) и затим показати како се разни сложенији поступци изражавају компоновањем библиотечких функционала над тако генерисаним листама. На пример, одређивање збира квадрата непарних цифара броја се може представити тако што се генерише серија цифара броја која се затим филтрира коришћењем функционала filter тако да јој се издвоје само непарне цифре, затим се те цифре квадрирају применом функционала map и на крају се израчуна њихов збир коришћењем агрегације (у овом случају сумације). Упоредити са традиционалним императивним начином да се такви задаци решавају и истаћи декларативност оваквог приступа програмирању. Истаћи значај решавања проблема разлагањем на мање и једноставније потпроблеме и аспекте функционалне парадигме који омогућавају да се мањи делови лако уклопе у целину (композиционалност и лењост као основни „лепак" који омогућава склапање програма од једноставнијих функција).

Обрадити рекурзију као примитивни механизам контроле тока програма. Приказати примере рекурзивно дефинисаних функција и упоредити их са имплементацијама истих функција које користе библиотечке функционалне и функције вишег реда. Истаћи предности изражавања на вишем нивоу апстракције и сугерисати избегавање непосредних рекурзивних имплементација када год је то могуће.

Истакнути и проблеме са ефикасношћу који настају услед коришћења имутабилних структура података и лењог израчунавања и приказати неке могућности оптимизације функционалних програма.

У оквиру теме **Савремена вештачка интелигенција** истаћи свеприсутност система вештачке интелигенције у свакодневном животу, са посебним акцентом на оне засноване на машинском учењу. Увести појам вештачке интелигенције као општу област која се бави постизањем интелигентног понашања рачунара, које је налик људском. Навести да је машинско учење једна од грана вештачке интелигенције где се решавање интелигентног задатка врши кроз анализу података на основу којих алгоритам машинског учења бива обучаван да га реши. Ученицима, кроз дискусију, приближити историјске аспекте машинског учења и укратко истаћи дистинкцију на класичне и савремене методе – дискутовати предности савремених метода (*дубоког* учења) у погледу учења обележја (енг. *features*), насупрот ручном пројектовању обележја присутног код класичних метода. Ученицима укратко приближити историјске узроке пробоја савремених метода машинског учења, а пре свега истаћи доступност велике количине података погодних за машинско учење, као и доступност адекватних напредних процесора – пре свега графичких картица. Идентификовати период око 2012. године као тачку прелома између класичних и савремених метода (описати ImageNet скуп података, и AlexNet модел као један од најважнијих доминантних резултата метода дубоког учења); али нагласити да се оба приступа користе и да сваки има примену која му посебно погодује.

Инсистирати да ученик може да препозна примере најчешћих система машинског учења у свету око себе (детекција ознаке на таблицама возила, гласовни асистент, препоручени филм на стриминг сервису…), као и да за неки дати пример ученик може да утврди да ли представља систем заснован на машинском учењу или не (светло које се аутоматски укључује када човек приђе није систем заснован на машинском учењу; аутомобил који се сам паркира може, али и не мора бити систем машинског учења; савремени аутоматски преводилац јесте систем машинског учења).

Представити ученицима машинско учење из угла парадигме програмирања на основу података. Истаћи да је у овој парадигми најзначајнија припрема самих података и дизајн модела и алгоритама учења, а да се не спроводи значајно експлицитно програмирање инструкција за решавање датог задатка. Као додатан пример савремених програмских парадигми кроз демонстрацију и вежбу илустровати систем машинског учења који аутоматски генерише стандардни рачунарски код на основу задатка писаног природним говором. Дискутовати са ученицима какве импликације на друштво, науку и технологију има наведени пример као и целокупна парадигме програмирања на основу података.

Дефинисати појмове тренинг и тест скупа и укратко дискутовати неопходност за њихову дисјунктност. Ученицима представити процес прикупљања, обраде и означавања података као често најзахтевнији и најскупљи елемент креирања система машинског учења. На примерима објаснити неопходност ручног означавања података (нпр. да би машина научила да детектује лица на слици, неопходно је дати јој примере слика на којима је човек већ означио где су лица), стимулисати ученике да сами предложе примере и дискутују тежину, односно цену њиховог означавања (рецимо, сегментационо означавање медицинских слика је јако скупо јер тај посао морају да раде лекари специјалисти). С обзиром да су анотатори (лабелари) људи који спроводе ручно означавање података, илустровати проблем њиховог неслагања на неким карактеристичним улазима нпр. за препознавање објеката или при обради природних језика. Полазећи од познатих практичних примера, илустровати редове величина скупова података неопходних за успешно обучавање савременог система машинског учења. Дефинисати појмове надгледаног машинског учења, машинског учења са поткрепљивањем и ненадгледаног машинског учења. Дефинисати опште класе задатака које решава модел машинског учења, а пре свега задатке класификације, задатке регресије и задатке кластеризације – илустровати ове класе и на примерима. Дефинисати најчешће моделе машинског учења, а посебно истаћи линеарну регресију, перцептрон и плитке неуралне мреже. Код илустровања рада перцептрона и неуралних мрежа начинити паралелу са биолошким нервним ћелијама. Дефинисати математички модел линеарног неурона, а онда објаснити неопходност увођења нелинеарности.

Навести парадигму дубоког учења као главни пример савременог машинског учења. На примеру неуралне мреже увести појам дубоког учења упоређујући га са сличностима и разликама класичних неуралних мрежа – истаћи разлику у дубини и броју слојева, количини података неопходних за обуку, а посебно обратити пажњу на разлику између обележја научених из података и ручно пројектованих обележја. Направити јасну разлику између параметара и хиперпараметара. Кроз дискусију са ученицима постићи интуитивно разумевање дубоког учења, без улажења у детаље имплементације, а посебно нагласити разлике у обележјима које уче нижи и виши слојеви (идеално кроз визуелизације) на једноставном примеру (рецимо детекција мачке на слици).

Кроз вежбе на рачунару омогућити ученицима да коришћењем готовог, већ истренираног, модела направе апликацију која решава неки интелигентно захтеван проблем (прављење апликације за детекцију лица на слици, прављење апликације за класификацију текста…). Фокусирати се на то да у изабраном програмском језику ученици могу да на готов модел повежу и на адекватан начин представе неопходне улазе, изврше модел и излазе правилно интерпретирају и прикажу. Препоручује се коришћење програмског језика који је ученицима већ познат, а у зависности од могућности

и афинитета ученика, наставник може увести *Python* као пример језика који се тренутно најчешће користи за машинско учење.

Увести основне статистичке метрике квалитета рада модела машинског учења које се користе за регресионе и класификационе моделе – средња апсолутна грешка, средња квадратна грешка, тачност, поновљивост, матрица конфузије и сл. Истаћи значај тренинг и тест скупа у контексту метрика квалитета рада модела машинског учења. Ученицима представити примере већ измерених резултата за неке конкретне моделе, дискутовати интерпретацију тих резултата, а посебно у функцији дате примене (рецимо, повишен ниво лажно позитивних предикција је велики проблем за систем који аутоматски пише казну за вожњу жутом траком, док је мање значајан за резултате тестирања на присуство заразног вируса). Може се и кроз вежбе на рачунару проћи имплементација основних статистичких метрика за већ дате резултате.

На примерима дводимензионалног скупа означених података детаљније дискутовати проблем класификације као пример примене парадигме програмирања на основу података. Кроз вежбе на рачунару омогућити да ученици сами имплементирају модел *k* најближих суседа за класификацију и анализирају његову успешност на илустративним скуповима података. Ученици могу и сами формирати скупове података за тренирање и тестирање, као и проћи кроз процедуру њиховог означавања. Може се проћи кроз исту материју и за вишедимензионе скупове података, а за амбициозније и кроз алгоритам *k* средина.

Када су ученици упознати са теоријским основама, практичном коришћењу готових модела, самосталном имплементирању једноставног модела као и мерењу квалитета модела може се приступити увођењу напреднијих практичних и етичких аспеката употребе машинског учења.

На конкретним примерима ученицима уводити практичне проблеме употребе машинског учења. Посебно истаћи и илустровати дефиницију преприлагођавања (енг. *overfitting*) и подприлагођавања (енг. *underfitting*); затим могућност постојања доменске разлике (енг. *domain gap*) између тренирајуће/тестирајућег скупа података и података у реалној експлоатацији модела; хардверске и енергетске проблеме имплементације модела у пракси; проблеме интерпретабилности код модела у критичним применама итд.

Посебно дискутовати са ученицима етичке проблеме употребе машинског учења. Илустровати класичне примере етичких недостатака модела, а онда навести ученике да сами предложе и дискутују могуће етичке проблеме у различитим гранама примене. Овде посебно образложити и проблеме приватности. Дискусијом и дебатом унутар одељења навести ученике да размишљају о потенцијалним законодавним решењима за етичке проблеме примене машинског учења.

III. ПРАЋЕЊЕ И ВРЕДНОВАЊЕ НАСТАВЕ И УЧЕЊА

У процесу вредновања потребно је континуирано пратити рад ученика. У настави оријентисаној на достизање исхода вреднују се и процес и продукти учења. Прикупљање информација из различитих извора (свакодневна посматрања, активност на часу, учествовање у разговору и дискусији, самосталан рад, рад у групи, тестови) помаже наставнику да сагледа постигнућа (развој и напредовање) ученика и степен остварености исхода. Свака активност је добра прилика за процену напредовања и давање повратне информације. Важно је и ученике оспособљавати и охрабривати да процењују сопствени напредак у учењу.

У процесу праћења и вредновања значајну улогу имају домаћи задаци. Редовно задавање домаћих задатака (уз обавезну повремену проверу од стране наставника), помаже наставнику да стекне бољи увид у степен остварености исхода кроз анализу задатака које ученици нису умели да реше. Важно је и мотивисати ученике који редовно раде домаће задатке тако што ће њихов рад бити оцењен.

Вредновање активности у оквиру тимског рада се може обавити са групом тако да се од сваког члана тражи објашњење елемената урађеног рада и мишљење о сопственом раду унутар тима. Препоручује се да наставник са ученицима договори показатеље на основу којих сви могу да прате напредак у учењу, ученици се уче да размишљају о квалитету свог рада и о томе шта треба да предузму да би свој рад унапредили. Оцењивање тако постаје инструмент за напредовање у учењу. На основу резултата праћења и вредновања, заједно са ученицима треба планирати процес учења и бирати погодне стратегије учења.

Препоручено је да коначна оцена за сваког ученика буде добијена комбиновањем различитих начина оцењивања:

– активност на часу, учествовање у разговору и дискусији;
– редовна израда домаћих задатака;
– тестови – провера знања;
– пројектни рад, и појединачни и тимски.

Комбиновање различитих начина оцењивања помаже да се сагледају слабе и јаке стране сваког ученика. Приликом сваког вредновања постигнућа потребно је ученику дати повратну информацију која помаже да разуме грешке и побољша свој резултат и учење. Потребно је да наставник резултате вредновања постигнућа својих ученика континуирано анализира и користи тако да промени део своје наставне праксе.

ВЕБ ПРОГРАМИРАЊЕ

Циљ учења предмета Веб програмирање је да ученици, кроз упознавање са савременим веб-технологијама и њихово коришћење у циљу креирања веб-садржаја који одговарају савременим пословним и личним потребама корисника, развију компетенције за рад са подацима и одговорно коришћење информационо-комуникационих технологија, као и да ученике оспособи за примену усвојених знања из области рачунарства и информатике, решавање разноврсних проблема из животне праксе, да их припреми за наставак образовања, да допринесе развијању менталних способности, формирању научног погледа на свет и свестраном развитку личности ученика.

ОПШТА ПРЕДМЕТНА КОМПЕТЕНЦИЈА

Учењем наставног предмета Веб програмирање ученици су оспособљен да примене стечена знања и вештине из области информационо-комуникационих технологија ради испуњавања постављених циљева и задатака у свакодневном животу, даљем школовању и будућем раду. Ученици развијају способност апстрактног и критичног мишљења уз помоћ информационо-комуникационих технологија. Развијају дигиталну писменост и позитивне ставове према рачунарским наукама.

СПЕЦИФИЧНЕ ПРЕДМЕТНЕ КОМПЕТЕНЦИЈЕ

Специфичне предметне компетенције представљају опис специфичних способности ученика које му омогућавају да развије општу предметну компетенцију. Ученици одговорно користе информационо-комуникационе технологије уз препознавање потенцијалних ризика и опасности. Брзо, ефикасно и рационално проналазе информације коришћењем рачунара, критички их анализирају и представљају у базама података. Познају концепте веб програмирања и језике за опис садржаја, изгледа и понашања веб страна. На основу познавања језика HTML и CSS, тумаче елементе веб-странице, прилагођавају их и креирају визуелно допадљиве странице које садрже линкове, слике, листе, табеле и сличне елементе. Овладали су вештинама и техникама неопходним за креирање сложенијих вишеслојних веб–апликација које омогућавају кориснику да кроз формулар уноси одређене податке, добије жељене податке, претражује, ажурира или брише податке из базе података.