Predtekmovanje. Pri predtekmovanju je bil cilj naloge doseči čim boljši rezultat predikcije o prihodih avtobusov LPP za eno linijo, v mesecu decembru. Za predikcijo smo morali uporabiti linearno regresijo. Osredotočali smo se predvsem na pravilno reprezentacijo podatkov in na filtriranje le-teh.
Podatke sem pripravil s pomočjo tehnike one-hot-encodinga. Le-ta se je izkazala za najboljšo možnost, zato sem jo uporabil tudi kasneje na tekmovanju.
Podatki so predstavljeni v 1D seznamu ničel in enk, kjer prvih 23 pol predstavlja uro v dnevu, naslednjih 7 dan v tednu, zadnji stolpec pa je predstavljal praznik oziroma dela prost dan. Zdi se mi, da mi je ta atribut na koncu nekoliko pokvaril rezultat, zato sem ga izpustil pred končno oddajo.
Vektorji z binarnimi vrednostmi, dolžine 31, predstavljajo kodirane podatke primerov voženj. En vektor predstavlja torej eno vožnjo, zadnji stolpec ene dimenzije pa je vseboval vse čase za korespondenčne vožnje.

Tako strukturirane podatke sem podal tako Linearnemu modelu, ki ga ponuja knjižnica Scikit learn, kot tudi funkciji linear.py, ki smo jo dobili kot gradivo in nam služi za izračun linearnega modela.
Slednja implementacija je bila bolj učinkovita od tiste iz scikit knjižnice, zato sem se v nadeljevanju raje ukvarjal z njo. Scikit model sem vsakič vseeno zgradil, da sem na njemu enostavno poklical funkcijo .score(), ki mi je podala nek objektiven rezultat, kako dober model je bil zgrajen. To oceno sem dobil na podlagi testiranja na učnih podatkih s prečnim preverjanjem (kar ni idealno, je pa "vsaj nekaj"). Tako sem lažje ugotovil, če ciljam v pravo smer, ali sem s kakšno značilko predikcijo pokvaril. Scikit za to točnost uporablja koeficient R^2 predikcije na učnih podatkih.
Regresije s polinomsko razrešitvijo nisem uporabil zaradi slabih iskušenj vrstnikov. Pa tudi samo z navadno lin. regresijo sem dobil dovolj dobre rezultate, zato po tej metodi tu nisem videl potrebe.
Pomembno se mi zdi še izpostaviti, da sem učil model le na zimskih mesecih, kar je bila moja prva ideja. Namreč na hitrost vožnje avtobusov predvsem vplivajo vremenske razmere, ki se močno razlikujejo po letnih časih. Po več testih se je izkazalo, da je najbolje trenirati model le na mesecu novembru in januarju, kajti te dva meseca sta najbolj podobna decembru - mesecu, v katerem so se nahajali vsi naši testni podatki predtekmovanja.

V veliko pomoč pri preprocesiranju podatkov mi je pomagala podana knjižnica lpputils.py.
V kodi je nekaj komentarjev (upam), ki so do neke mere relavantni. Upam, da bodo v kakšno pomoč.

Tekmovanje. Pri tekmovanju sem uporabil vse trike, ki sem jih odkril že v predtekmovanju, pa še mnogo več.
Z vremenom se nisem preveč ukvarjal. podatke sem si sicer pripravil v cvs datoteko za padavine, vendar mi rezultatov ni izboljšalo. Prav tako sem jih zakodiral v stolpec za dež in sneg, kjer je vrednost 1 predstavljala prisotnost, 0 pa odsotnost določenega tipa padavin, a to kmalu opustil.
Glavni trik, implementacije pri tekmovanju je bil, da sem za vsako linijo zgradil svoj napovedovalni model, katere sem potem uporabljal pri napovedovanju na ustreznih mestih.
Še ena glavnih izboljšav je bila filtracija t.i. outlierjev. To so podatki, ki imajo nenormalne vrednosti. Ti so mi v učnih podatkih kvarili modele predikcij, zato sem jih izločil. Take izjeme sem ujel v funkciji join_day_time(), ki je originalno združila le dan in čas, od tod tako ime, potem pa sem dodajal še minute, praznike,.. ime pa je ostalo.
Ujel sem jih tako, da sem vse primere, ko se je avtobus peljal le nekaj sekund ali pa več ur (dni), enostavno preskolčil pri grajenju učne množice.

Uporabil sem tudi izboljšane podatke o dela prostih dnevih v letu 2012 ter šolskih počitnicah (v mesecu decembru) in med drugim zasledil zanimivo objavo na spletu:
'Obveščamo vas, da bodo v času šolskih počitnic, od 24. 12. do 31. 12. 2012, z izjemo linij 3, 3B, 3G, 15, 18, 19B, 19I, 25 in 27 veljali počitniški vozni redi.'
To sem ustrezno upošteval pri atributu, ki je opisoval primer glede na to, ali je vozil po počitniškem voznem redu ali ne.

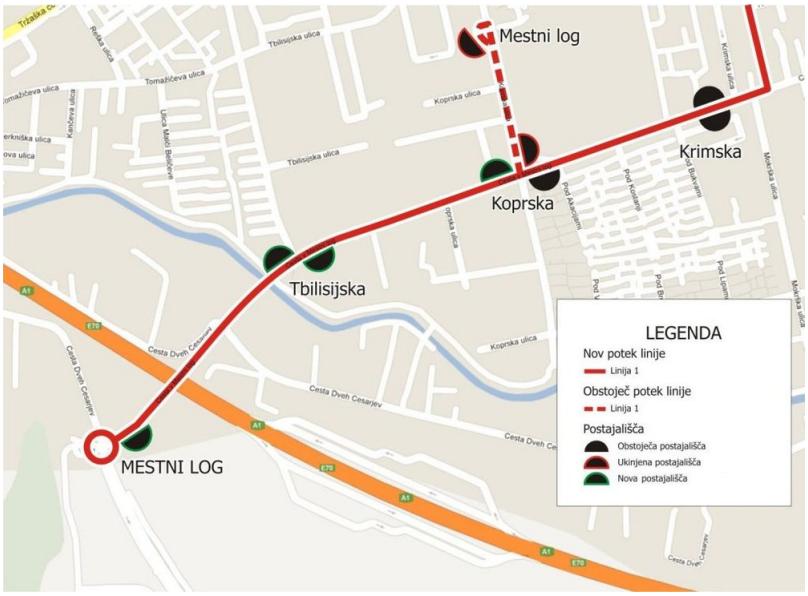
Nekatere linije so imele več različnih končnih postaj, zato sem modele gradil še bolj strogo, tudi na podlagi te informacije.
Tu je prišlo do manjšega zapleta, kajti v testnih podatki so se pojavile linije s končnimi postajami, za katere modeli niso bili zgrajeni , zaradi pomanjkanja primernih učnih podatkov. To sem rešil tako, da sem takim linijam določil "najbližji"možni model. Primer take linije je avtobus na relaciji "MESTNI LOG - VIŽMARJE", z označbo 1.
Na spletu je v letu 2012 krožil tudi članek, ki govori o tem, da se je zgoraj omenjeni liniji (1) 3.12.2012 začela vozit do druge končne postaje. To lahko vidimo na sliki 1 spodaj.

Pri mojem najboljšem rezultatu so vektorje primerov sestavljali vektorji dolgi 33 enot. Prvih 23 prostorov kodira 24 ur, naslednji 3 prostori zakodirajo četrtine ur (zato, da mi ni bilo potrebno ustvariti 60 novih stolpcev, za vsako minuto posebaj), naslednjih 6 stolpcev je opisovalo 7 dni v tednu, zadnji stolpec pa še počitnice. Primer, kako sem zakodiral uro na 15-minutno natančnost:

```
000 - 0-14 minut
100 - 15-29 minut
010 - 30-44 minut
001 - 45-59 minut
```

Na podoben način sem zakodiral tudi ostale vrednosti in tako prišparal povsod po 1 stolpec, kar je malo izboljšalo modele napram implementaciji v predtekmovanju.

Primer funkcije, kjer zakodiram TESTNE podatke, preden nad njimi izvedem predikcijo:



Slika 1: Spremenjena končna postaja avtobusa številke 1.

```
def encode_example(example, sola):
    """
    :param example: not-encoded example
    :return: one-hot-encoded example
    """
    example_encoded = [0] * 33
    departure = lpp.parsedate(example[6])  # departure time without 0's at the end
    linija = int(example[2])
    smer = example[3][0]

    if departure.hour != 0:
        example_encoded[departure.hour - 1] = 1  # first 0-22 slots are for deprature hours
    if (departure.minute // 15) != 0:
        example_encoded[((departure.minute // 15) - 1) + 23] = 1  # 23, 24 and 25th slot for
    if departure.isoweekday() != 1:
        example_encoded[(departure.isoweekday() - 2) + 26] = 1  # 26-31 slots are for days.

    zimske_pocitnice = ["24,12", "25,12", "26,12", "27,12", "28,12", "29,12", "30,12", "31,12
    datum = "" + str(departure.day) + "," + str(departure.month)
    if [datum] in sola:
        example_encoded[32] = 1

    # set holidays back to 0 if the route is immune to winter holidays
    imune_linije = ["3", "3B", "3G", "15", "18", "19B", "19I", "25", "27"]
    if datum in zimske_pocitnice:
        if linija == 3 and smer in ("L", "B", "G"): # preveri ce je linija imuna na pocitnice
            example_encoded[32] = 0
        elif linija == 19:
            example_encoded[32] = 0
        elif linija in (15, 18, 25, 27):
            example_encoded[32] = 0

    return example_encoded
```

Če sem kje kakšen trik izpustil, je verjetno lepo raztviden iz kode, po možnsti pa še zako-

mentiran.
Pri računanju R^2 vrednosti sem tu zadevo rešil tako, da sem ocenil R^2 vrednost na vsakem modelu za pripadajočo linijo, jih seštel in povprečil. Tako sem dobil nek približek ocene modela.
Cel algoritem prebere podane CSV podatke, jih ustrezno filtrira in zakodira, da se na njih lahko zgradijo linearni modeli s pomočjo linear.py, prebere testne podatke, vsak primer zakodira v ustrezno obliko, naredi napoved nad modelom in na koncu vse zbrane napovedi zapiše v izhodno datoteko.

Končni rezultat me je dvignil okrog 20-tega mesta, s čimer sem zadovoljen.
Prilagam še sliki mojih rezultatov iz predtekmovanja in tekmovanja.

ID	Rezultat	Čas oddaje	Ista oddaja na lestvici	Število oddaj za lestvico	Ime datoteke
L	?	2020-12-11 05:41:30	165.90893	9	rez.txt
L	?	2020-12-11 05:30:04	165.90913	9	rez.txt
L	?	2020-12-11 01:11:19	167.01143	9	rez1.txt
L	?	2020-12-11 00:37:35	168.53700	9	rez1.txt
L	?	2020-12-10 22:42:58	172.25965	9	rez.txt
L	?	2020-12-11 00:01:30	185.45925	9	rez.txt
L	?	2020-12-10 23:46:01	192.40527	9	rez.txt
L	?	2020-12-10 23:46:23	192.40527	9	rez.txt
L	?	2020-12-10 23:49:23	194.98540	9	rez.txt

Slika 2: Rezultati tekmovanja.

ID	Rezultat	Čas oddaje	Ista oddaja na lestvici	Število oddaj za lestvico	Ime datoteke
L	?	2020-12-04 02:36:38	147.33783	11	rez11_no_holidays.txt
L	?	2020-12-04 02:26:05	147.55085	11	rez10_work_free_days_linear.txt
L	?	2020-12-01 23:44:38	147.61655	11	rez4.txt
L	?	2020-12-02 23:42:12	148.53284	11	rez6_vikendi.txt
L	?	2020-12-01 22:52:34	149.11965	11	predtekmovanje1_ju.txt
L	?	2020-12-01 23:40:05	152.57759	11	rez3.txt
L	?	2020-12-01 23:37:19	155.16084	11	rez2.txt
L	?	2020-12-02 23:41:34	183.20270	11	rez8_sklearn_dnevi.txt
L	?	2020-12-04 02:13:59	183.78209	11	rez9_dela_prosti_dnevi.txt
L	?	2020-12-02 23:42:23	183.79279	11	rez7_sklearn_vikendi.txt
L	?	2020-12-04 02:40:30	185.63063	11	rez11_no_holidays_sklearn.txt

Slika 3: Rezultati predtekmovanja.