



# گزارش تمرین سوم درس شبکه‌های عصبی

فاطمه غلامزاده

۹۹۱۳۱۰۰۳



## سوال اول

با بررسی مجموعه داده مشخص شد که ۶ مورد از ستون horsepower با علامت “?” پر شده‌اند. برای جایگزینی این موارد راه‌های متعددی وجود دارند که چند مورد را بررسی می‌کنیم:

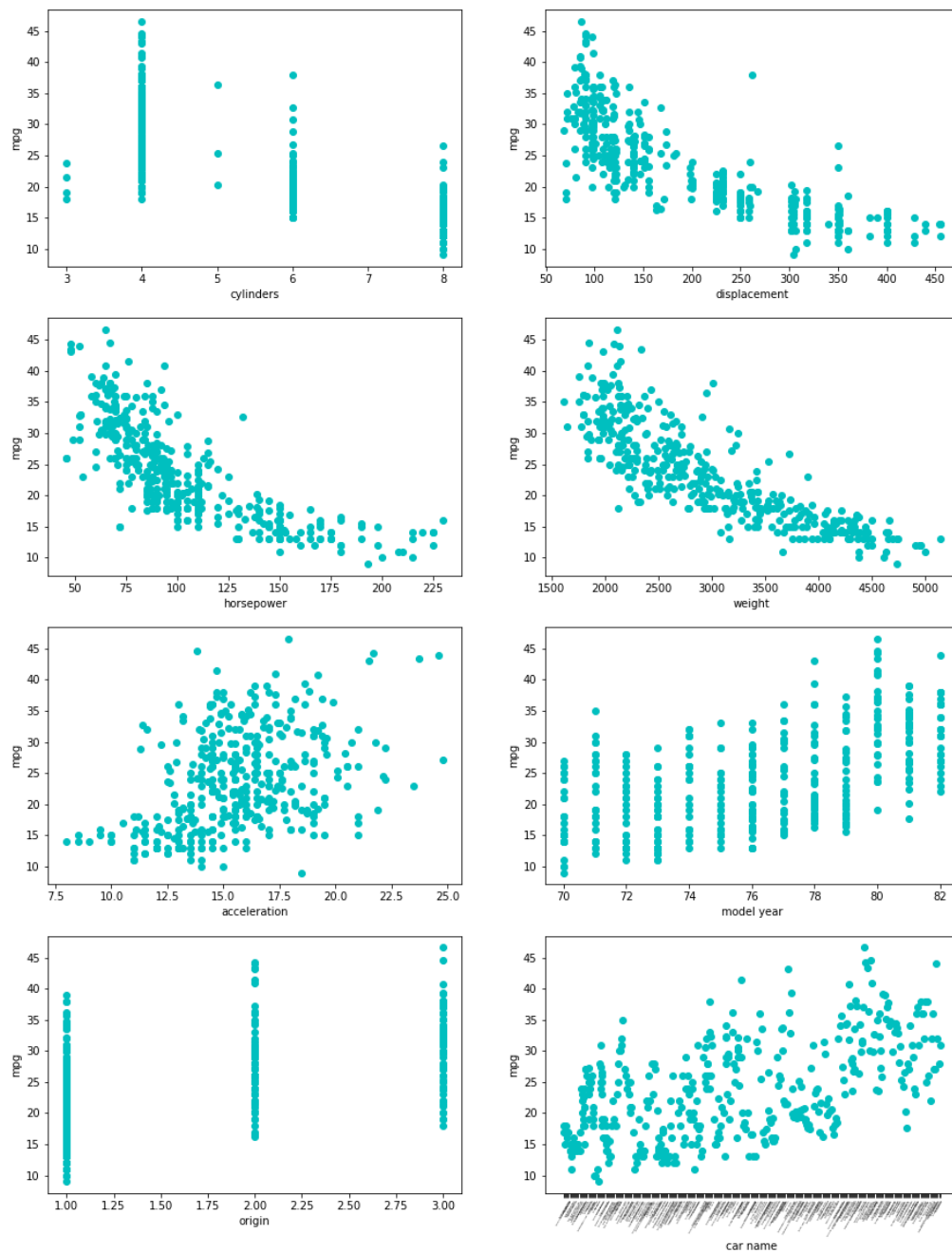
- ۱- **جایگزینی با میانگین (mean):** در این روش مقادیر گمشده را با میانگین سایر مقادیر آن ستون پر می‌کنیم. این روش سریع است اما میانگین در برابر داده‌های نویزی مقاوم نیست و همچنین برای فیچرهای categorical نمی‌تواند به کار گرفته شود. همچنین استفاده از داده‌هایی که مقادیر گمشده‌شان با میانگین پر شده است ممکن باعث وجود بایاس زیاد در مدل شوند.
- ۲- **استفاده از مُد (mode):** در این روش مقادیر گمشده را با پرتکرارترین عنصر ستون مربوطه پر می‌کنیم. این روش نیز از آنجایی که همبستگی‌ها را در نظر نمی‌گیرد می‌تواند موجب بایاس زیاد در مدل شود.
- ۳- **استفاده از میانه (median):** در این روش مقادیر گمشده را با میانه داده‌های آن ستون جایگزین می‌کنیم. این روش نسبت به داده‌های نویزی مقاوم است.
- ۴- **استفاده از interpolation:** در این روش یک تابع به مقادیر موجود در آن ستون که مقادیر گمشده دارد fit می‌شود. می‌توان از interpolation‌های خطی یا چندجمله‌ای استفاده کرد. از نظر محاسباتی از دو روش اول پیچیده‌تر است.
- ۵- **استفاده از KNN:** در این روش k نزدیکترین همسایه‌ای که بیشترین شباهت را به داده‌ای که مقدار گمشده دارد می‌یابیم و به کمک آن‌ها مقدار گمشده را پر می‌کنیم. برای هر سه نوع داده‌ی گسسته، پیوسته و categorical کاربرد دارد.
- ۶- **استفاده از deep learning:** این روش برای فیچرهای categorical و غیر عددی خیلی خوب عمل می‌کند و از روش‌های یادگیری عمیق برای جایگزینی مقادیر گمشده استفاده می‌کند.

در این سوال مقادیر گمشده با روش میانه (median) جایگزین شدند زیرا این روش برای دیتاست‌های کوچک مثل این دیتاست خوب جواب می‌دهد. همچنین تعداد مقادیر گمشده نسبت به اندازه دیتاست خیلی کم است و

فقط حدود ۱,۵ درصد است. از طرفی روش میانه نسبت به داده‌های نویزی مقاوم است بنابراین این روش انتخاب شد و ۶ داده‌ی گم‌شده با 93.5 که مقدار میانه ستون horsepower است جایگزین شدند.

## سوال دوم

شکل زیر نمودارهای میزان مصرف سوخت به ازای هر یک از ویژگی‌ها را نشان می‌دهد.



شکل ۱: نمودار میزان مصرف سوخت به ازای هر ویژگی

## سوال سوم

### بخش اول: kmeans

ساختار کلی شبکه‌های این سوال به این صورت است:

- **لایه ورودی:** در این لایه از ۷ نورون استفاده شده است زیرا تعداد نورون‌ها در لایه ورودی برابر است با تعداد ویژگی‌ها یا به عبارت دیگر ابعاد داده‌های ورودی. با توجه به اینکه داده‌های مورد استفاده در این سوال، ۷ ویژگی دارند در لایه ورودی به ۷ نورون نیاز داریم. این نورون‌ها هیچ پردازشی روی داده ورودی انجام نمی‌دهند و فقط داده را به شبکه وارد می‌کنند. به همین دلیل در لایه ورودی نیازی به تعریف تابع فعالیت نیست.
- **لایه خروجی:** در این لایه از یک نورون استفاده شده است. زیرا یک مساله رگرسیون داریم و می‌خواهیم مقدار یک متغیر را پیش‌بینی کنیم.

- **لایه RBF:** تعداد نورون‌های این لایه همان تعداد مراکز توابع گاوسی است که در این مسئله متغیر است و تغییرات آن را بررسی می‌کنیم. توابع فعالیت نورون‌ها در این لایه در واقع همان توابع گاوسی هستند.

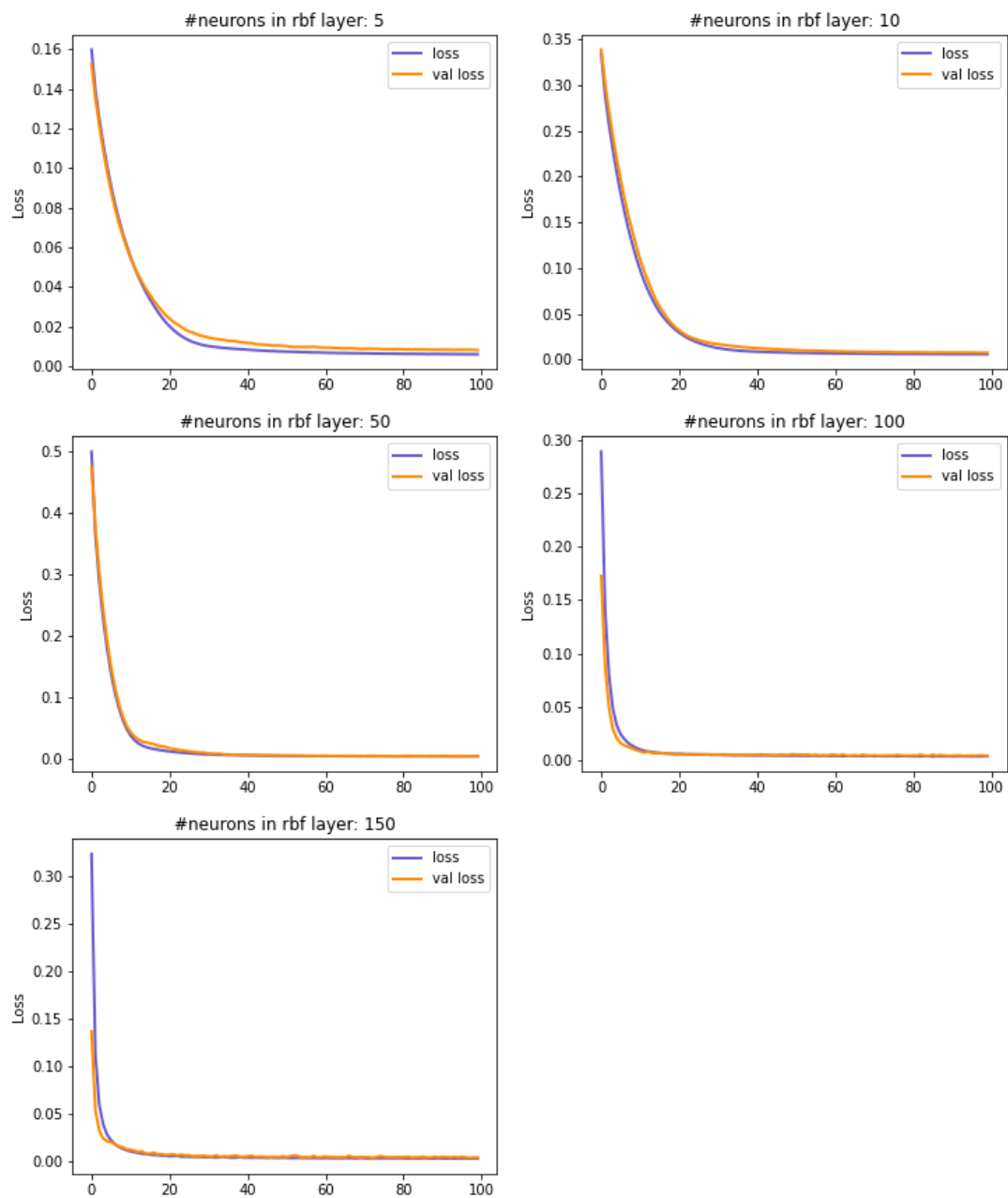
همچنین در کامپایل مدل از RMSprop به عنوان optimizer و از mse برای loss استفاده شده است.

در جدول زیر خطاهای MSE به دست آمده روی داده‌های تست، به ازای تعداد نورون‌های مختلف در لایه RBF گزارش شده است. همانطور که مشاهده می‌شود تعداد نورون ۱۰۰ کمترین خطا را دارد.

با افزایش تعداد نورون‌ها ابتدا عملکرد مدل بهبود یافته است و سپس رو به بدتر شدن می‌رود. زیرا در ابتدا با تعداد نورون‌های کم در واقع تعداد مراکز توابع گاوسی کم است و مدل خیلی ساده است و نمی‌تواند به خوبی تابع را تخمین بزند (underfitting). سپس با زیاد شدن تعداد نورون‌ها پیچیدگی مدل به حد مطلوبی می‌رسد که در آنجا کمترین خطا را داریم. بعد مجدداً با افزایش تعداد نورون‌ها عملکرد دوباره رو به بدتر شدن می‌رود، زیرا پیچیدگی مدل زیاد می‌شود و استعداد overfitting پیدا می‌کند.

MSE	تعداد نورون‌ها در لایه RBF
0.0046	5
0.0039	10
0.0035	50
0.0030	100
0.0036	150

در شکل ۲ نمودارهای loss و val\_loss برحسب iteration برای تعداد لایه‌های مختلف رسم شده است.



شکل ۲ نمودار loss بر حسب تعداد تکرار به ازای تعداد نوروں مختلف در لایه مخفی

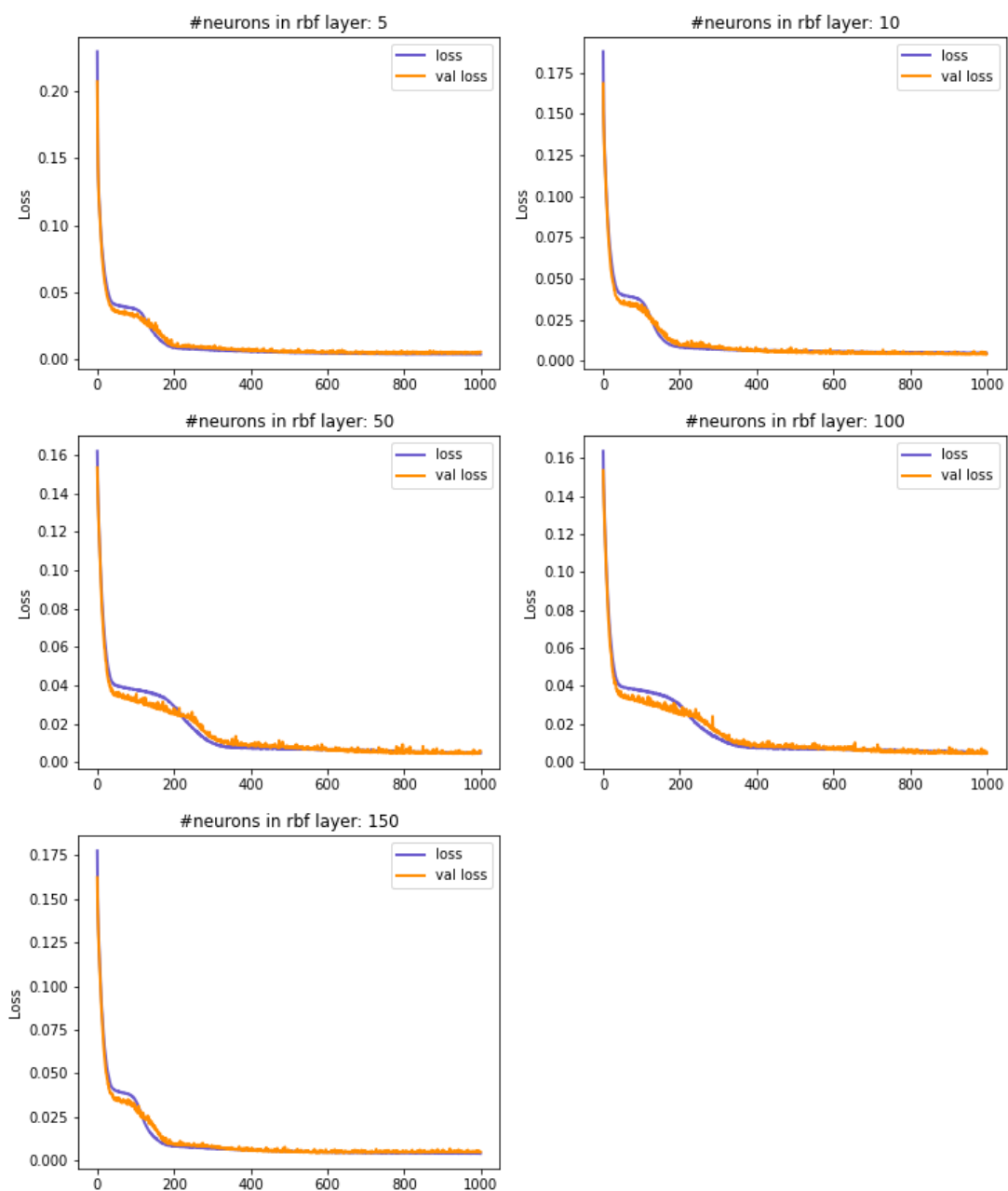
## بخش دوم: OLS

در جدول زیر خطاهای MSE به دست آمده روی داده‌های تست، به ازای تعداد نورون‌های مختلف در لایه RBF و استفاده از روش OLS برای انتخاب مراکز، گزارش شده است. همانطور که مشاهده می‌شود تعداد نورون ۱۵۰ کمترین خطا را دارد.

همانطور که دیده می‌شود مقدار خطا با افزایش تعداد نورون‌ها ابتدا روند نزولی، سپس روند صعودی و بعد دوباره روند نزولی دارد و با انجام آزمایشات متعدد مشخص شد که روند مشخصی ندارد. علت آن می‌تواند این باشد که انتخاب مراکز توابع گاوسی برای ساختن ماتریس P در روش OLS به صورت تصادفی صورت می‌گیرد و موجب این روند نامشخص می‌شود اما نکته ای که وجود دارد این است که این خطاها اختلاف خیلی کمی با هم دارند و نزدیک به هم هستند و همه‌ی آن‌ها مقادیر نسبتاً کمی دارند که مناسب است.

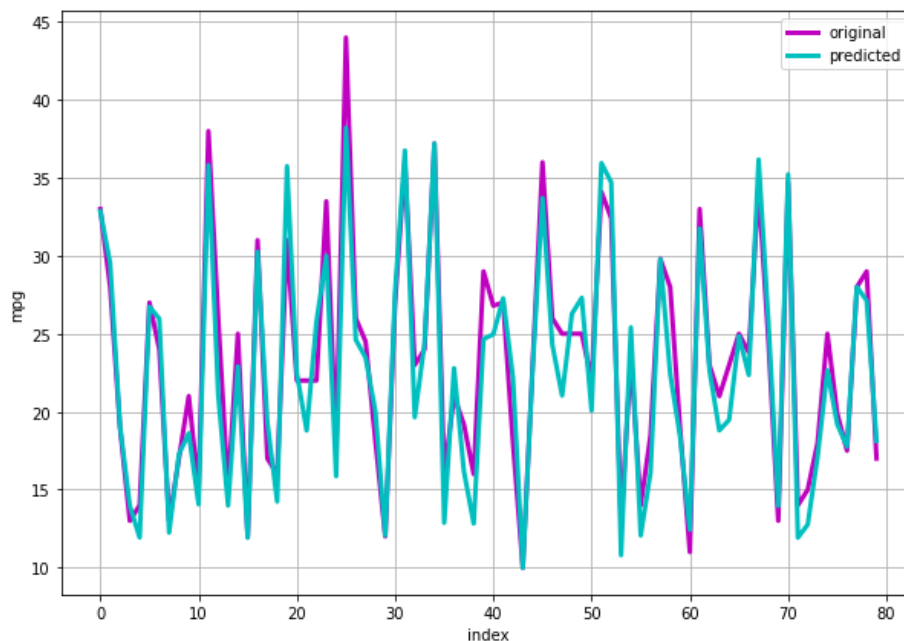
MSE	تعداد نورون‌ها در لایه‌ی RBF
0.0042	5
0.0036	10
0.0053	50
0.0044	100
0.0033	150

در شکل ۳ نمودارهای loss و val\_loss برحسب iteration برای تعداد لایه‌های مختلف رسم شده است.



شکل ۳ نمودار loss بر حسب تعداد تکرار به ازای تعداد نوروں های مختلف در لایه مخفی

بنابراین بهترین مدل RBF یافت شده دارای لایه مخفی با 100 نورون و الگوریتم kmeans برای یافتن مراکز می باشد. شکل ۴ نمودار مقادیر واقعی و پیش بینی شده توسط این مدل برای متغیر mpg را نشان می دهد.



شکل ۴: نمودار مقادیر واقعی و پیش بینی شده توسط شبکه RBF با kmeans برای متغیر mpg

### مقایسه عملکرد OLS و Kmeans:

همانطور که مشاهده می شود عملکرد این شبکه ها اختلاف چندانی با هم ندارد اما در حالتی که با استفاده از kmeans مراکز انتخاب می شوند عملکرد کمی بهتر است. علت آن می تواند این باشد که روش OLS برای دیتاست های بزرگ بهتر عمل می کند. دلیل دیگر آن می تواند این باشد که روش OLS بر اساس رگرسیون عمل می کند و سعی دارد بردارهای رگرسوری که بر هم عمود هستند را بیابد. رگرسیون یک روش خطی است، ینی در OLS این فرض ضمنی را داریم که داده ها رابطه خطی دارند در حالی که در kmeans اینگونه نیست و فرض ضمنی خطی بودن داده ها را نداریم.

## سوال چهارم

ساختار کلی شبکه‌های این سوال به این صورت است:

- **لایه ورودی:** در این لایه از ۷ نورون استفاده شده است زیرا تعداد نورون‌ها در لایه ورودی برابر است با تعداد ویژگی‌ها یا به عبارت دیگر ابعاد داده‌های ورودی. با توجه به اینکه داده‌های مورد استفاده در این سوال، ۷ ویژگی دارند در لایه ورودی به ۷ نورون نیاز داریم. این نورون‌ها هیچ پردازشی روی داده ورودی انجام نمی‌دهند و فقط داده را به شبکه وارد می‌کنند. به همین دلیل در لایه ورودی نیازی به تعریف تابع فعالیت نیست.
  - **لایه خروجی:** در این لایه از یک نورون استفاده شده است. زیرا یک مساله رگرسیون داریم و می‌خواهیم مقدار یک متغیر را پیش‌بینی کنیم.
  - **لایه پنهان:** در این لایه از تابع فعالیت Relu استفاده شده است. تعداد لایه‌ها، تعداد نورون‌های هر لایه و نرخ یادگیری با آزمایشاتی که در هر بخش از این سوال خواسته شده تعیین شده است.
- همچنین در کامپایل مدل از adam به عنوان optimizer و از mse برای loss استفاده شده است.

## بخش اول

در جدول زیر تعداد لایه‌های مخفی و خطایی که هر مدل با آن تعداد لایه دارد، مشخص شده است. در هر لایه مخفی تعداد نورون‌ها ۲۰ در نظر گرفته شده است. همانطور که مشخص است تعداد لایه ۳ کمترین خطا را دارد. افزایش تعداد لایه‌ها در ابتدا باعث کاهش خطا شده است اما از تعداد لایه ۳ به بعد با افزایش تعداد لایه‌ها مقدار خطا نیز افزایش پیدا می‌کند که یکی از دلایل آن می‌تواند overfit شدن مدل به دلیل زیاد شدن پیچیدگی باشد. در ابتدا که تعداد لایه‌ها خیلی کم است مدل خیلی ساده است و simplicity آن بالاست که اصطلاحاً گفته می‌شود underfitting داریم. در ادامه با افزایش تعداد لایه‌ها به یک مدل مناسب می‌رسیم که کمترین خطا را دارد و بعد مجدداً با افزایش تعداد لایه‌ها خطا افزایش می‌یابد زیرا پیچیدگی مدل زیاد می‌شود و یک مدل complex داریم که احتمال overfit شدن آن بالاست.

تعداد لایه	مقدار خطای MSE
1	0.0053
2	0.0044
3	0.0038
5	0.0040
10	0.0047
50	0.0047
100	0.0389

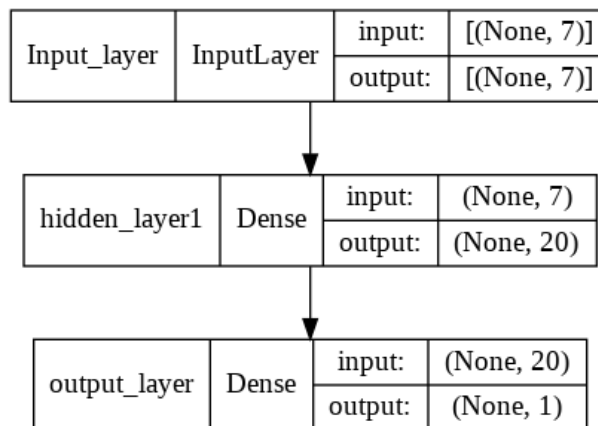


مشخصات مدل برای مدلی با یک عدد لایه مخفی در زیر آورده شده است:

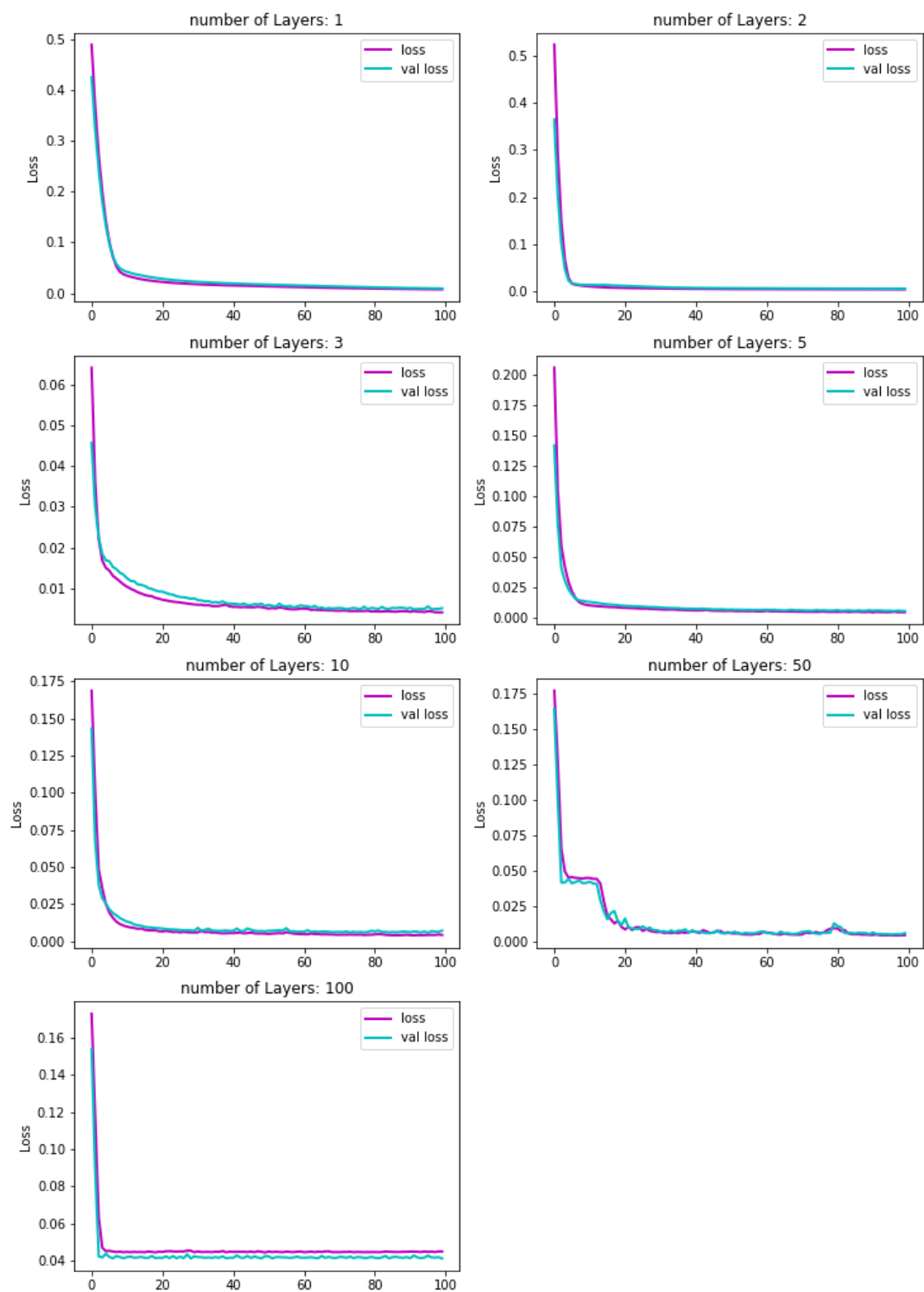
Model: "sequential\_49"

Layer (type)	Output Shape	Param #
hidden_layer1 (Dense)	(None, 20)	160
output_layer (Dense)	(None, 1)	21

=====  
Total params: 181  
Trainable params: 181  
Non-trainable params: 0



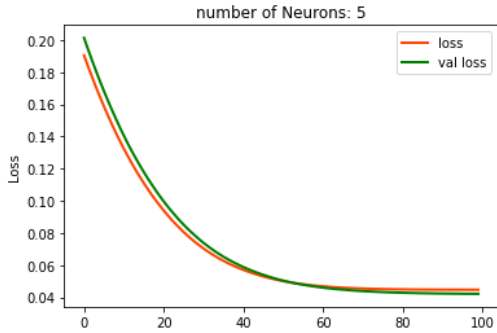
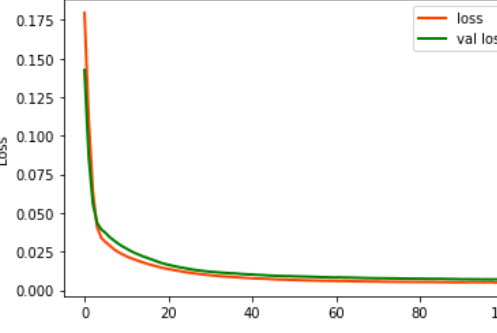
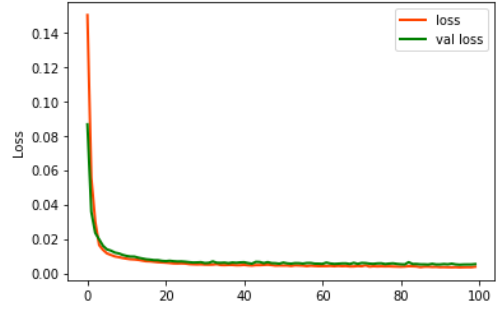
در شکل ۲ نمودارهای loss و val\_loss برحسب iteration برای تعداد لایه‌های مختلف رسم شده است.

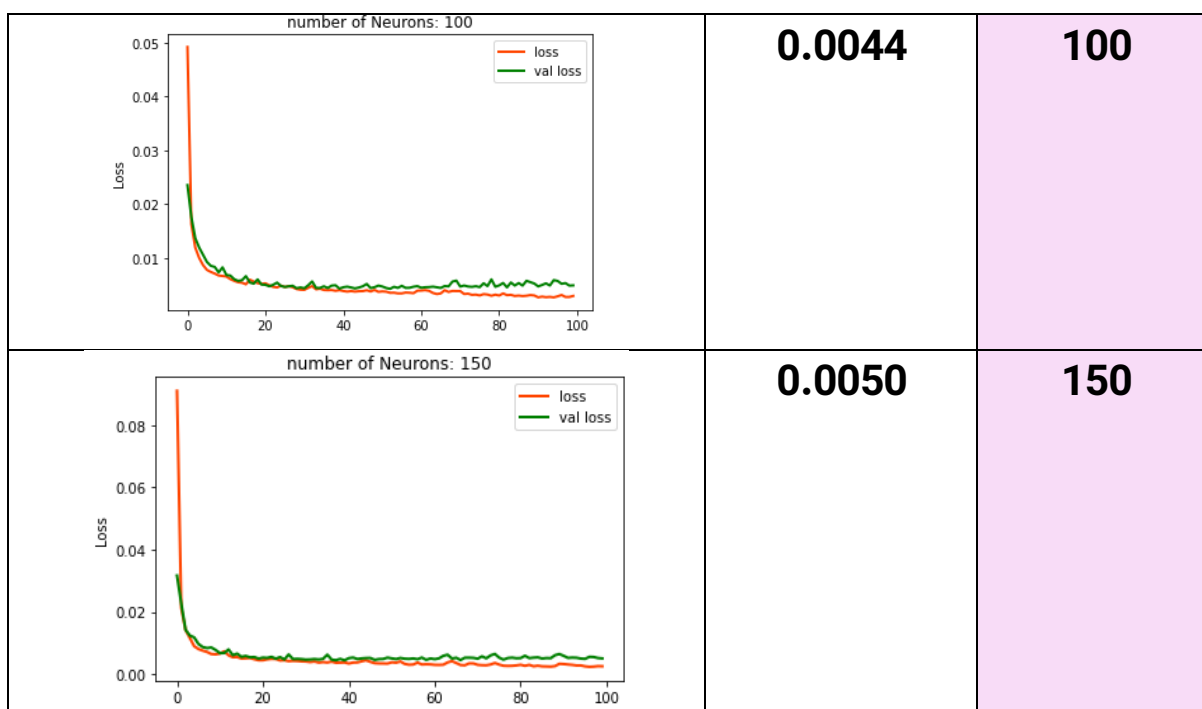


شکل ۵: نمودار loss بر حسب تعداد تکرار به ازای تعداد لایه‌های مختلف

## بخش دوم)

بهترین تعداد لایه‌ها که از بخش قبلی به دست آمد برابر ۳ است. در جدول زیر تعداد لایه‌های نورون‌های مختلف با در نظر گرفتن ۳ لایه مخفی و خطایی که هر مدل با آن تعداد نورون دارد، مشخص شده است. همانطور که دیده می‌شود تعداد نورون ۵۰ کمترین خطا را دارد. افزایش تعداد نورون‌ها در ابتدا باعث کاهش خطا شده است اما از تعداد نورون ۵۰ به بعد با افزایش تعداد نورون‌ها مقدار خطا نیز افزایش پیدا می‌کند که یکی از دلایل آن می‌تواند overfit شدن مدل به دلیل زیاد شدن پیچیدگی باشد. در واقع در ابتدا مدل تا حد زیادی ساده است و بایاس آن بالاست و underfit می‌شود، سپس به یک مدل مناسب می‌رسیم و بعد دوباره با افزایش تعداد نورون‌ها پیچیدگی مدل زیاد می‌شود و واریانس و استعداد overfit شدن مدل بالا می‌رود.

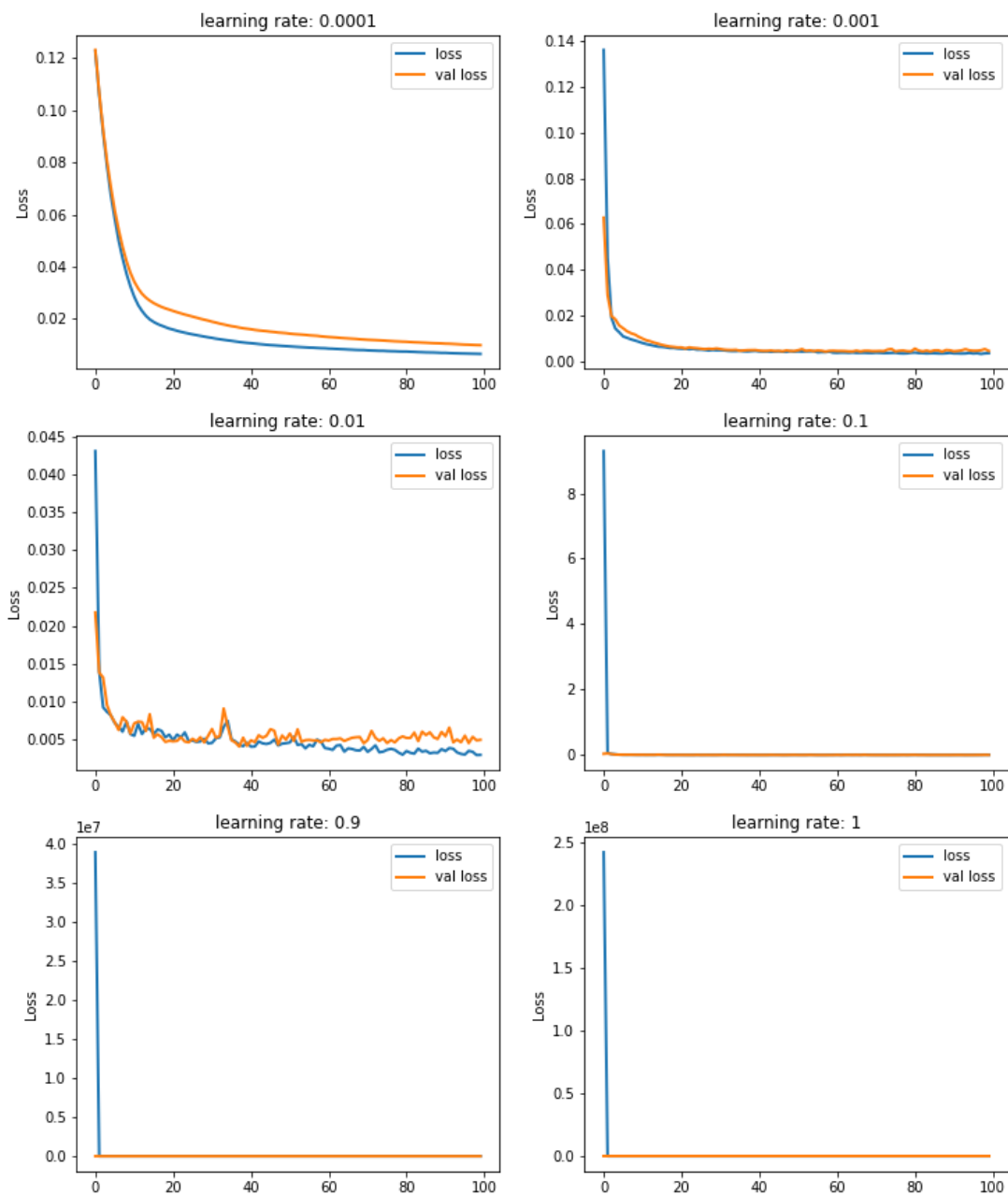
تعداد لایه	MSE	نمودار loss بر حسب iteration
5	0.0381	
10	0.0037	
50	0.0035	



### بخش سوم)

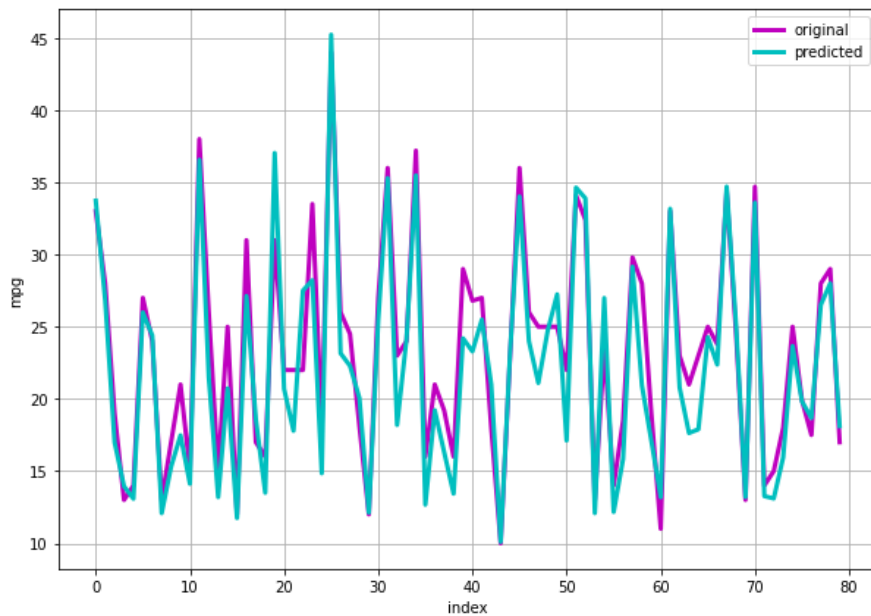
بهترین تعداد لایه‌ها که از بخش قبلی به دست آمد برابر ۳ و بهترین تعداد نورون ۵۰ است. در جدول زیر مقدار نرخ‌های آموزش مختلف با در نظر گرفتن ۳ لایه مخفی و ۵۰ نورون و خطایی که هر مدل دارد، مشخص شده است. همانطور که دیده می‌شود  $\text{learning rate} = 0.01$  کمترین خطا را دارد. افزایش نرخ یادگیری در ابتدا باعث کاهش خطا شده است اما از نرخ یادگیری ۰.۰۱ به بعد با افزایش نرخ یادگیری مقدار خطا نیز افزایش پیدا می‌کند. علت این امر این است که وقتی نرخ یادگیری پایین است با قدم‌های کوچک به سمت بهینه سراسری گام برمیداریم و ممکن است به آن نرسیم یا اینکه به اندازه کافی به آن نزدیک نشویم. از طرفی وقتی نرخ یادگیری خیلی بالا باشد آنقدر قدم‌ها بزرگ است که به احتمال زیاد بهینه سراسری را از دست می‌دهیم و ممکن است حتی به آن نزدیک هم نشویم. در شکل ۳ نمودارهای loss بر حسب iteration برای نرخ‌های یادگیری مختلف نشان داده شده است.

MSE	Learning rate
0.0058	0.0001
0.0040	0.001
0.0035	0.01
0.0058	0.1
0.2124	0.9
9.3891	1



شکل 6 نمودار loss بر حسب تعداد تکرار برای نرخ‌های یادگیری مختلف

بنابراین بهترین مدل MLP یافت شده دارای ۳ لایه مخفی با ۵۰ نورون و مقدار نرخ یادگیری 0.01 می‌باشد. شکل ۷ نمودار مقادیر واقعی و پیش‌بینی شده برای متغیر mpg را نشان می‌دهد.



شکل ۷ مقادیر واقعی و پیش بینی شده توسط شبکه MLP برای متغیر mpg

## سوال پنجم

همانطور که در نتایج سوال ۳ و ۴ دیده می شود خطای بهترین شبکه MLP برای 0.0035 و خطای بهترین شبکه RBF برابر 0.0030 به دست آمد. بنابراین در مورد مجموعه داده این تمرین، شبکه RBF عملکرد بهتری داشت و خطای MSE کمتری را تولید کرد.

شبکه RBF و MLP از نظر ساختار شبیه به هم هستند و هر دو از لایه ورودی، خروجی و لایه(های) مخفی تشکیل شده اند. هم چنین هر دوی آنها در دسته ی شبکه های feed forward قرار دارند.

در یک مسئله پیچیده که داده ها جداپذیر خطی نیستند یک راه حل این است که یک تبدیل غیرخطی به داده ها اعمال کنیم و آنها را به فضایی با ابعاد بزرگتر مساوی فضای فعلی ببریم و این احتمال وجود دارد که این مسئله در این فضای جدید جدایی پذیر خطی شود. این ایده در شبکه ی RBF به کار گرفته شده است.

در شبکه RBF ما می توانیم یک تابع را با یک سری توابع دیگر تخمین بزنیم که به آنها توابع پایه شعاعی گفته می شود. در واقع عمل تبدیل به فضای جدید در لایه ی مخفی شبکه RBF که به لایه RBF نیز معروف است انجام می گیرد. RBF دو نوع پارامتر را یاد می گیرد:

۱. مراکز و عرض توابع پایه شعاعی
۲. وزن هایی که توابع پایه باید با هم ترکیب شوند تا تابع خروجی را بسازند

اولین مجموعه از پارامترها را می توان به طور مستقل از مجموعه دوم پارامترها یاد گرفت. با انجام خوشه بندی K-means، می توانید مراکز و عرض RBF ها را بیابید.

همانطور که گفته شد شبکه RBF ابعاد بردارهای ویژگی را افزایش می دهد و این باعث می شود که جدایی پذیری آن ها افزایش پیدا کند. در شبکه RBF همانند MLP می توانیم از الگوریتم BP یا ترکیبی از الگوریتم های دیگر برای آموزش شبکه استفاده کنیم.

- یک حسن شبکه RBF نسبت به MLP این است که محاسبات پیچیده ای ندارد و نسبتاً سریع است. خود شبکه نیز ساده است و بهینه سازی برای لایه RBF و لایه خروجی به صورت جداگانه انجام می گیرد.
- مزیت دیگر RBF نسبت به MLP این است که تابعی که هر نود در لایه RBF دارد به راحتی قابل تفسیر است اما در MLP این تفسیرپذیری را نداریم.
- در RBF تعداد لایه های مخفی و تعداد نورون ها در لایه مخفی راحت تر از MLP تعیین می شود.
- اما یک مزیتی که MLP نسبت به RBF دارد این است که در مسائل دسته بندی بهتر از RBF عمل می کند و در زمان کمتری پاسخ می دهد.
- شبکه های RBF برای تشخیص الگوهای جدید مناسبند اما برای برون یابی (extrapolation) مناسب نیستند. هم چنین در RBF اضافه کردن نورون های جدید در طول آموزش راحت تر است.