Walmart Kaggle Contest

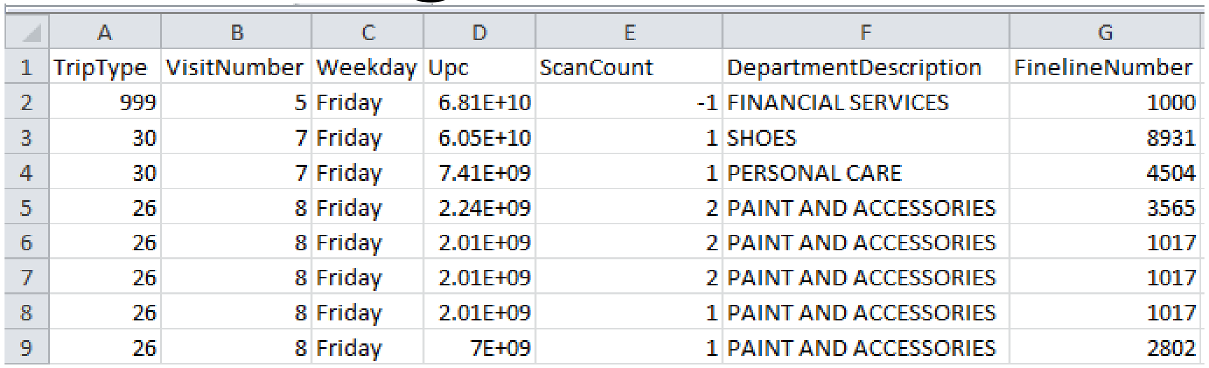
[Trip Type Classification]

Predict “Trip Type” starting with 6 Features

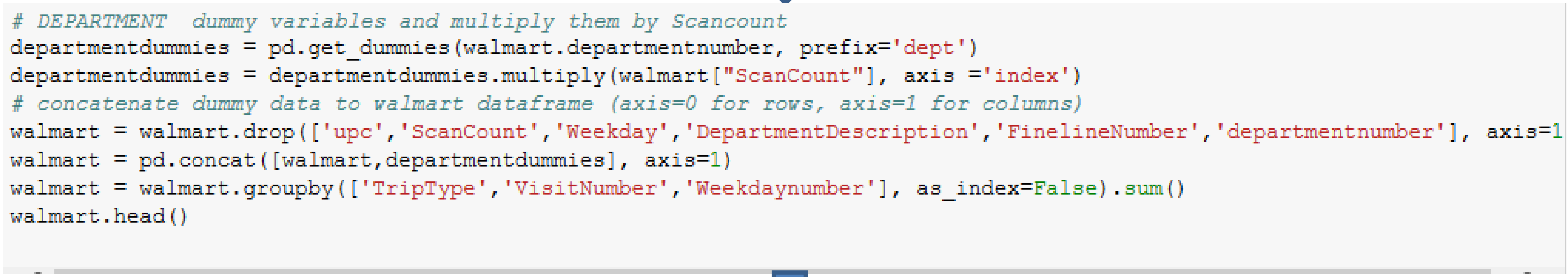
(Trip Type: categorical id representing the type of shopping trip the customer made)

1. **VisitNumber** - an id corresponding to a single trip by a single customer
2. **Weekday** - the weekday of the trip
3. **Upc** - the UPC number of the product purchased
4. **ScanCount** - the number of the given item that was purchased. A negative value indicates a product return.
5. **DepartmentDescription** - a high-level description of the item's department
6. **FinelineNumber** - a more refined category for each of the products, created by Walmart

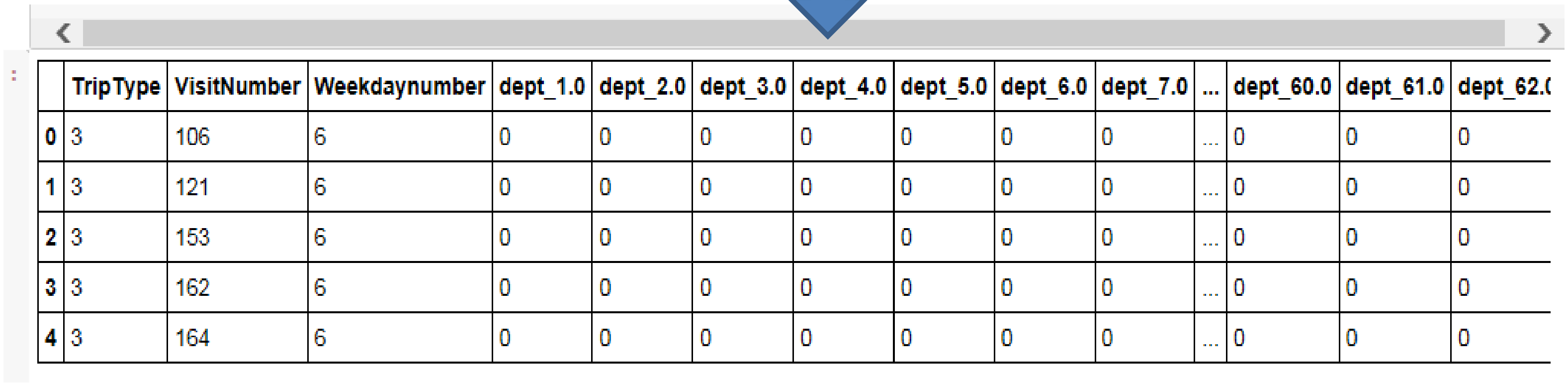
Working with the Data



	A	B	C	D	E	F	G
1	TripType	VisitNumber	Weekday	Upc	ScanCount	DepartmentDescription	FinelineNumber
2	999	5	Friday	6.81E+10	-1	FINANCIAL SERVICES	1000
3	30	7	Friday	6.05E+10	1	SHOES	8931
4	30	7	Friday	7.41E+09	1	PERSONAL CARE	4504
5	26	8	Friday	2.24E+09	2	PAINT AND ACCESSORIES	3565
6	26	8	Friday	2.01E+09	2	PAINT AND ACCESSORIES	1017
7	26	8	Friday	2.01E+09	2	PAINT AND ACCESSORIES	1017
8	26	8	Friday	2.01E+09	1	PAINT AND ACCESSORIES	1017
9	26	8	Friday	7E+09	1	PAINT AND ACCESSORIES	2802



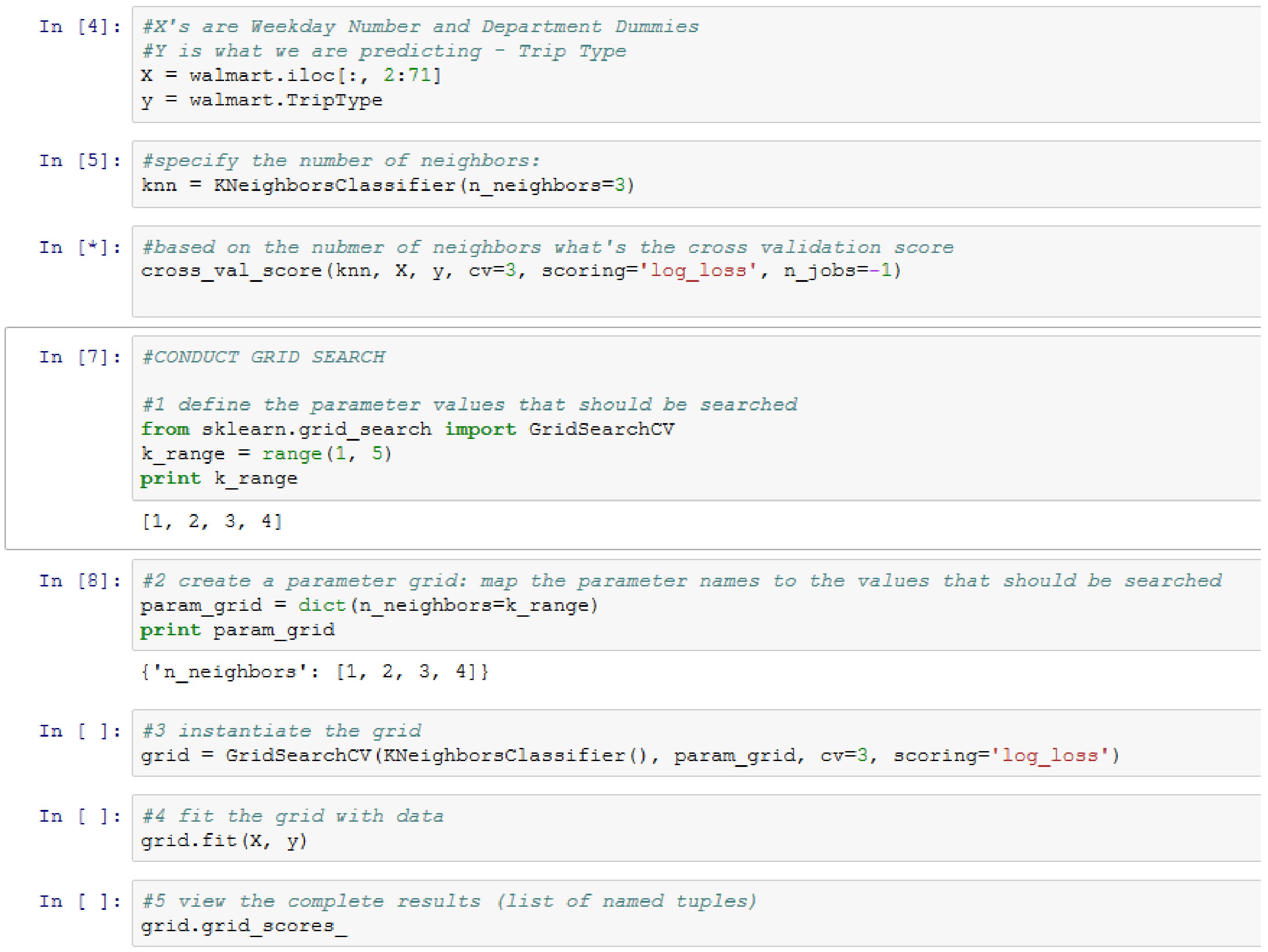
```
# DEPARTMENT  dummy variables and multiply them by Scancount
departmentdummies = pd.get_dummies(walmart.departmentnumber, prefix='dept')
departmentdummies = departmentdummies.multiply(walmart["ScanCount"], axis ='index')
# concatenate dummy data to walmart dataframe (axis=0 for rows, axis=1 for columns)
walmart = walmart.drop(['upc','ScanCount','Weekday','DepartmentDescription','FinelineNumber','departmentnumber'], axis=1
walmart = pd.concat([walmart,departmentdummies], axis=1)
walmart = walmart.groupby(['TripType','VisitNumber','Weekdaynumber'], as_index=False).sum()
walmart.head()
```



	TripType	VisitNumber	Weekdaynumber	dept_1.0	dept_2.0	dept_3.0	dept_4.0	dept_5.0	dept_6.0	dept_7.0	...	dept_60.0	dept_61.0	dept_62.(
0	3	106	6	0	0	0	0	0	0	0	...	0	0	0
1	3	121	6	0	0	0	0	0	0	0	...	0	0	0
2	3	153	6	0	0	0	0	0	0	0	...	0	0	0
3	3	162	6	0	0	0	0	0	0	0	...	0	0	0
4	3	164	6	0	0	0	0	0	0	0	...	0	0	0

Modeling

- K Nearest Neighbor to predict Trip Type



```
In [4]: #X's are Weekday Number and Department Dummies
        #Y is what we are predicting - Trip Type
        X = walmart.iloc[:, 2:71]
        y = walmart.TripType

In [5]: #specify the number of neighbors:
        knn = KNeighborsClassifier(n_neighbors=3)

In [*]: #based on the nubmer of neighbors what's the cross validation score
        cross_val_score(knn, X, y, cv=3, scoring='log_loss', n_jobs=-1)

In [7]: #CONDUCT GRID SEARCH

        #1 define the parameter values that should be searched
        from sklearn.grid_search import GridSearchCV
        k_range = range(1, 5)
        print k_range

        [1, 2, 3, 4]

In [8]: #2 create a parameter grid: map the parameter names to the values that should be searched
        param_grid = dict(n_neighbors=k_range)
        print param_grid

        {'n_neighbors': [1, 2, 3, 4]}

In [ ]: #3 instantiate the grid
        grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=3, scoring='log_loss')

In [ ]: #4 fit the grid with data
        grid.fit(X, y)

In [ ]: #5 view the complete results (list of named tuples)
        grid.grid_scores_
```

Next Steps

1. Find optimal model through grid search
2. Load Test set and run against model
3. Export results to Kaggle competition